VISIT…

Ingemar J. Cox
Matthew L. Miller
Jeffrey A. Bloom
Jessica Fridrich
Ton Kalker

Digital Watermarking *and* Steganography

MORGAN KAUFMANN

SECOND EDITION

Digital Watermarking and Steganography

The Morgan Kaufmann Series in Multimedia Information and Systems

Series Editor, Edward A. Fox, Virginia Poytechnic University

Digital Watermarking and Steganography, Second Edition
Ingemar J. Cox, Matthew L. Miller, Jeffrey A. Bloom, Jessica Fridrich, and Ton Kalker

Keeping Found Things Found: The Study and Practice of Personal Information Management
William P. Jones

Web Dragons: Inside the Myths of Search Engine Technology
Ian H. Witten, Marco Gori, and Teresa Numerico

Introduction to Data Compression, Third Edition
Khalid Sayood

Understanding Digital Libraries, Second Edition
Michael Lesk

Bioinformatics: Managing Scientific Data
Zoé Lacroix and Terence Critchlow

How to Build a Digital Library
Ian H. Witten and David Bainbridge

Readings in Multimedia Computing and Networking
Kevin Jeffay and Hong Jiang Zhang

Multimedia Servers: Applications, Environments, and Design
Dinkar Sitaram and Asit Dan

Visual Information Retrieval
Alberto del Bimbo

Managing Gigabytes: Compressing and Indexing Documents and Images, Second Edition
Ian H. Witten, Alistair Moffat, and Timothy C. Bell

Digital Compression for Multimedia: Principles & Standards
Jerry D. Gibson, Toby Berger, Tom Lookabaugh, Rich Baker, and David Lindbergh

Readings in Information Retrieval
Karen Sparck Jones, and Peter Willett

For further information on these books and for a list of forthcoming titles, please visit our web site at http://www.mkp.com.

The Morgan Kaufmann Series in Computer Security

Digital Watermarking and Steganography, Second Edition
Ingemar J. Cox, Matthew L. Miller, Jeffrey A. Bloom, Jessica Fridrich, and Ton Kalker

Information Assurance: Dependability and Security in Networked Systems
Yi Qian, David Tipper, Prashant Krishnamurthy, and James Joshi

Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS
Jean-Philippe Vasseur, Mario Pickavet, and Piet Demeester

For further information on these books and for a list of forthcoming titles, please visit our Web site at http://www.mkp.com.

Digital Watermarking and Steganography

Second Edition

Ingemar J. Cox

Matthew L. Miller

Jeffrey A. Bloom

Jessica Fridrich

Ton Kalker

ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Morgan Kaufmann Publishers is an imprint of Elsevier

MORGAN KAUFMANN PUBLISHERS

Publishing Director	Denise E. M. Penrose
Senior Acquisitions Editor	Rick Adams
Publishing Services Manager	George Morrison
Senior Project Manager	Brandy Lilly
Editorial Assistant	Gregory Chalson
Cover Design	Dennis Schaefer
Text Design	Elsevier, Inc.
Composition	diacriTech
Copyeditor	Janet Cocker
Proofreader	Jodie Allen
Indexer	Distributech Scientific Indexing
Interior printer	The Maple-Vail Book Manufacturing Group
Cover printer	Phoenix Color

Morgan Kaufmann Publishers is an imprint of Elsevier.
30 Corporate Drive, Suite 400, Burlington, MA 01803, USA

This book is printed on acid-free paper. ♾

Copyright © 2008 by Elsevier Inc. All rights reserved.

Designations used by companies to distinguish their products are often claimed as trademarks or registered trademarks. In all instances in which Morgan Kaufmann Publishers is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means—electronic, mechanical, photocopying, scanning, or otherwise—without prior written permission of the publisher.

Permissions may be sought directly from Elsevier's Science & Technology Rights Department in Oxford, UK: phone: (+44) 1865 843830, fax: (+44) 1865 853333, E-mail: permissions@elsevier.com. You may also complete your request online via the Elsevier homepage (http://elsevier.com), by selecting "Support & Contact" then "Copyright and Permission" and then "Obtaining Permissions."

Library of Congress Cataloging-in-Publication Data
Digital watermarking and steganography/Ingemar J. Cox ... [et al.].
p. cm.
Includes bibliographical references and index.
ISBN 978-0-12-372585-1 (casebound: alk. paper) 1. Computer security. 2. Digital watermarking. 3. Data protection. I. Cox, I. J. (Ingemar J.)
QA76.9.A25C68 2008
005.8–dc22

2007040595

ISBN 978-0-12-372585-1

For information on all Morgan Kaufmann publications,
visit our Web site at *www.mkp.com* or *www.books.elsevier.com*

Printed in the United States of America
07 08 09 10 11 5 4 3 2 1

Working together to grow
libraries in developing countries
www.elsevier.com | www.bookaid.org | www.sabre.org
ELSEVIER BOOK AID International Sabre Foundation

This book is dedicated to the memory of

Ingy Cox

Age 12

May 23, 1986 to January 27, 1999

The light that burns twice as bright burns half as long—and you have burned so very very brightly.

—Eldon Tyrell to Roy Batty in *Blade Runner*.
Screenplay by Hampton Fancher and David Peoples.

This page intentionally left blank

Contents

Preface to the First Edition xv
Preface to the Second Edition xix
Example Watermarking Systems xxi

CHAPTER 1 Introduction **1**
1.1 Information Hiding, Steganography, and Watermarking 4
1.2 History of Watermarking 6
1.3 History of Steganography 9
1.4 Importance of Digital Watermarking 11
1.5 Importance of Steganography 12

CHAPTER 2 Applications and Properties **15**
2.1 Applications of Watermarking 16
2.1.1 Broadcast Monitoring 16
2.1.2 Owner Identification 19
2.1.3 Proof of Ownership 21
2.1.4 Transaction Tracking 23
2.1.5 Content Authentication 25
2.1.6 Copy Control 27
2.1.7 Device Control 31
2.1.8 Legacy Enhancement 32
2.2 Applications of Steganography 34
2.2.1 Steganography for Dissidents 34
2.2.2 Steganography for Criminals 35
2.3 Properties of Watermarking Systems 36
2.3.1 Embedding Effectiveness 37
2.3.2 Fidelity 37
2.3.3 Data Payload 38
2.3.4 Blind or Informed Detection 39
2.3.5 False Positive Rate 39
2.3.6 Robustness 40
2.3.7 Security 41
2.3.8 Cipher and Watermark Keys 43
2.3.9 Modification and Multiple Watermarks 45
2.3.10 Cost 46
2.4 Evaluating Watermarking Systems 46
2.4.1 The Notion of “Best” 47
2.4.2 Benchmarking 47
2.4.3 Scope of Testing 48

2.5 Properties of Steganographic and Steganalysis Systems 49
2.5.1 Embedding Effectiveness 49
2.5.2 Fidelity 50
2.5.3 Steganographic Capacity, Embedding Capacity, Embedding Efficiency, and Data Payload 50
2.5.4 Blind or Informed Extraction 51
2.5.5 Blind or Targeted Steganalysis 51
2.5.6 Statistical Undetectability 52
2.5.7 False Alarm Rate 53
2.5.8 Robustness 53
2.5.9 Security 54
2.5.10 Stego Key 54
2.6 Evaluating and Testing Steganographic Systems 55
2.7 Summary 56

CHAPTER 3 Models of Watermarking 61
3.1 Notation 62
3.2 Communications 63
3.2.1 Components of Communications Systems 63
3.2.2 Classes of Transmission Channels 64
3.2.3 Secure Transmission 65
3.3 Communication-Based Models of Watermarking 67
3.3.1 Basic Model 67
3.3.2 Watermarking as Communications with Side Information at the Transmitter 75
3.3.3 Watermarking as Multiplexed Communications 78
3.4 Geometric Models of Watermarking 80
3.4.1 Distributions and Regions in Media Space 81
3.4.2 Marking Spaces 87
3.5 Modeling Watermark Detection by Correlation 95
3.5.1 Linear Correlation 96
3.5.2 Normalized Correlation 97
3.5.3 Correlation Coefficient 100
3.6 Summary 102

CHAPTER 4 Basic Message Coding 105
4.1 Mapping Messages into Message Vectors 106
4.1.1 Direct Message Coding 106
4.1.2 Multisymbol Message Coding 110
4.2 Error Correction Coding 117
4.2.1 The Problem with Simple Multisymbol Messages 117
4.2.2 The Idea of Error Correction Codes 118
4.2.3 Example: Trellis Codes and Viterbi Decoding 119

4.3 Detecting Multisymbol Watermarks 124
4.3.1 Detection by Looking for Valid Messages 125
4.3.2 Detection by Detecting Individual Symbols 126
4.3.3 Detection by Comparing against Quantized Vectors . . . 128
4.4 Summary 134

CHAPTER 5 Watermarking with Side Information 137
5.1 Informed Embedding 139
5.1.1 Embedding as an Optimization Problem 140
5.1.2 Optimizing with Respect to a Detection Statistic 141
5.1.3 Optimizing with Respect to an Estimate of Robustness 147
5.2 Watermarking Using Side Information 153
5.2.1 Formal Definition of the Problem 153
5.2.2 Signal and Channel Models 155
5.2.3 Optimal Watermarking for a Single Cover Work 156
5.2.4 Optimal Coding for Multiple Cover Works 157
5.2.5 A Geometrical Interpretation of White Gaussian Signals 158
5.2.6 Understanding Shannon's Theorem 159
5.2.7 Correlated Gaussian Signals 161
5.3 Dirty-Paper Codes 164
5.3.1 Watermarking of Gaussian Signals: First Approach 164
5.3.2 Costa's Insight: Writing on Dirty Paper 170
5.3.3 Scalar Watermarking 175
5.3.4 Lattice Codes 179
5.4 Summary 181

CHAPTER 6 Practical Dirty-Paper Codes 183
6.1 Practical Considerations for Dirty-Paper Codes 183
6.1.1 Efficient Encoding Algorithms 184
6.1.2 Efficient Decoding Algorithms 185
6.1.3 Tradeoff between Robustness and Encoding Cost 186
6.2 Broad Approaches to Dirty-Paper Code Design 188
6.2.1 Direct Binning 188
6.2.2 Quantization Index Modulation 188
6.2.3 Dither Modulation 189
6.3 Implementing DM with a Simple Lattice Code 189
6.4 Typical Tricks in Implementing Lattice Codes 194
6.4.1 Choice of Lattice 194
6.4.2 Distortion Compensation 194
6.4.3 Spreading Functions 195
6.4.4 Dither 195

6.5 Coding with Better Lattices . . . 197
6.5.1 Using Nonorthogonal Lattices . . . 197
6.5.2 Important Properties of Lattices . . . 199
6.5.3 Constructing a Dirty-Paper Code from E_8 . . . 201
6.6 Making Lattice Codes Survive Valumetric Scaling . . . 204
6.6.1 Scale-Invariant Marking Spaces . . . 205
6.6.2 Rational Dither Modulation . . . 207
6.6.3 Inverting Valumetric Scaling . . . 208
6.7 Dirty-Paper Trellis Codes . . . 208
6.8 Summary . . . 212

CHAPTER 7 Analyzing Errors 213
7.1 Message Errors . . . 214
7.2 False Positive Errors . . . 218
7.2.1 Random-Watermark False Positive . . . 219
7.2.2 Random-Work False Positive . . . 221
7.3 False Negative Errors . . . 225
7.4 ROC Curves . . . 228
7.4.1 Hypothetical ROC . . . 228
7.4.2 Histogram of a Real System . . . 230
7.4.3 Interpolation Along One or Both Axes . . . 231
7.5 The Effect of Whitening on Error Rates . . . 232
7.6 Analysis of Normalized Correlation . . . 239
7.6.1 False Positive Analysis . . . 240
7.6.2 False Negative Analysis . . . 250
7.7 Summary . . . 252

CHAPTER 8 Using Perceptual Models 255
8.1 Evaluating Perceptual Impact of Watermarks . . . 255
8.1.1 Fidelity and Quality . . . 256
8.1.2 Human Evaluation Measurement Techniques . . . 257
8.1.3 Automated Evaluation . . . 260
8.2 General Form of a Perceptual Model . . . 263
8.2.1 Sensitivity . . . 263
8.2.2 Masking . . . 266
8.2.3 Pooling . . . 267
8.3 Two Examples of Perceptual Models . . . 269
8.3.1 Watson's DCT-Based Visual Model . . . 269
8.3.2 A Perceptual Model for Audio . . . 273
8.4 Perceptually Adaptive Watermarking . . . 277
8.4.1 Perceptual Shaping . . . 280
8.4.2 Optimal Use of Perceptual Models . . . 287
8.5 Summary . . . 295

CHAPTER 9 Robust Watermarking 297
9.1 Approaches . . . 298
9.1.1 Redundant Embedding . . . 299
9.1.2 Spread Spectrum Coding . . . 300
9.1.3 Embedding in Perceptually Significant Coefficients . . . 301
9.1.4 Embedding in Coefficients of Known Robustness . . . 302
9.1.5 Inverting Distortions in the Detector . . . 303
9.1.6 Preinverting Distortions in the Embedder . . . 304
9.2 Robustness to Valumetric Distortions . . . 308
9.2.1 Additive Noise . . . 308
9.2.2 Amplitude Changes . . . 312
9.2.3 Linear Filtering . . . 314
9.2.4 Lossy Compression . . . 319
9.2.5 Quantization . . . 320
9.3 Robustness to Temporal and Geometric Distortions . . . 325
9.3.1 Temporal and Geometric Distortions . . . 326
9.3.2 Exhaustive Search . . . 327
9.3.3 Synchronization/Registration in Blind Detectors . . . 328
9.3.4 Autocorrelation . . . 329
9.3.5 Invariant Watermarks . . . 330
9.3.6 Implicit Synchronization . . . 331
9.4 Summary . . . 332

CHAPTER 10 Watermark Security 335
10.1 Security Requirements . . . 335
10.1.1 Restricting Watermark Operations . . . 336
10.1.2 Public and Private Watermarking . . . 338
10.1.3 Categories of Attack . . . 340
10.1.4 Assumptions about the Adversary . . . 345
10.2 Watermark Security and Cryptography . . . 348
10.2.1 The Analogy between Watermarking and Cryptography . . . 348
10.2.2 Preventing Unauthorized Detection . . . 349
10.2.3 Preventing Unauthorized Embedding . . . 351
10.2.4 Preventing Unauthorized Removal . . . 355
10.3 Some Significant Known Attacks . . . 358
10.3.1 Scrambling Attacks . . . 359
10.3.2 Pathological Distortions . . . 359
10.3.3 Copy Attacks . . . 361
10.3.4 Ambiguity Attacks . . . 362
10.3.5 Sensitivity Analysis Attacks . . . 367
10.3.6 Gradient Descent Attacks . . . 372
10.4 Summary . . . 373

CHAPTER 11 Content Authentication 375
11.1 Exact Authentication . . . 377
11.1.1 Fragile Watermarks . . . 377
11.1.2 Embedded Signatures . . . 378
11.1.3 Erasable Watermarks . . . 379
11.2 Selective Authentication . . . 395
11.2.1 Legitimate versus Illegitimate Distortions . . . 395
11.2.2 Semi-Fragile Watermarks . . . 399
11.2.3 Embedded, Semi-Fragile Signatures . . . 404
11.2.4 Telltale Watermarks . . . 409
11.3 Localization . . . 410
11.3.1 Block-Wise Content Authentication . . . 411
11.3.2 Sample-Wise Content Authentication . . . 412
11.3.3 Security Risks with Localization . . . 415
11.4 Restoration . . . 419
11.4.1 Embedded Redundancy . . . 419
11.4.2 Self-Embedding . . . 420
11.4.3 Blind Restoration . . . 421
11.5 Summary . . . 422

CHAPTER 12 Steganography 425
12.1 Steganographic Communication . . . 427
12.1.1 The Channel . . . 428
12.1.2 The Building Blocks . . . 429
12.2 Notation and Terminology . . . 433
12.3 Information-Theoretic Foundations of Steganography . . . 433
12.3.1 Cachin's Definition of Steganographic Security . . . 434
12.4 Practical Steganographic Methods . . . 439
12.4.1 Statistics Preserving Steganography . . . 439
12.4.2 Model-Based Steganography . . . 441
12.4.3 Masking Embedding as Natural Processing . . . 445
12.5 Minimizing the Embedding Impact . . . 449
12.5.1 Matrix Embedding . . . 450
12.5.2 Nonshared Selection Rule . . . 457
12.6 Summary . . . 467

CHAPTER 13 Steganalysis 469
13.1 Steganalysis Scenarios . . . 469
13.1.1 Detection . . . 470
13.1.2 Forensic Steganalysis . . . 475
13.1.3 The Influence of the Cover Work on Steganalysis . . . 476
13.2 Some Significant Steganalysis Algorithms . . . 477
13.2.1 LSB Embedding and the Histogram Attack . . . 478

13.2.2 Sample Pairs Analysis 480
13.2.3 Blind Steganalysis of JPEG Images Using Calibration . . . 486
13.2.4 Blind Steganalysis in the Spatial Domain 489
13.3 Summary 494

APPENDIX A Background Concepts **497**
A.1 Information Theory 497
A.1.1 Entropy 497
A.1.2 Mutual Information 498
A.1.3 Communication Rates 499
A.1.4 Channel Capacity 500
A.2 Coding Theory 503
A.2.1 Hamming Distance 503
A.2.2 Covering Radius 503
A.2.3 Linear Codes 504
A.3 Cryptography 505
A.3.1 Symmetric-Key Cryptography 505
A.3.2 Asymmetric-Key Cryptography 506
A.3.3 One-Way Hash Functions 508
A.3.4 Cryptographic Signatures 510

APPENDIX B Selected Theoretical Results **511**
B.1 Information-Theoretic Analysis of Secure Watermarking (Moulin and O'Sullivan) 511
B.1.1 Watermarking as a Game 511
B.1.2 General Capacity of Watermarking 513
B.1.3 Capacity with MSE Fidelity Constraint 514
B.2 Error Probabilities Using Normalized Correlation Detectors (Miller and Bloom) 517
B.3 Effect of Quantization Noise on Watermarks (Eggers and Girod) . 522
B.3.1 Background 524
B.3.2 Basic Approach 524
B.3.3 Finding the Probability Density Function 524
B.3.4 Finding the Moment-Generating Function 525
B.3.5 Determining the Expected Correlation for a Gaussian Watermark and Laplacian Content 527

APPENDIX C Notation and Common Variables **529**
C.1 Variable Naming Conventions 529
C.2 Operators 530
C.3 Common Variable Names 530
C.4 Common Functions 532

Glossary 533

References 549

Index 575

About the Authors 591

Preface to the First Edition

Watermarking, as we define it, is the practice of hiding a message about an image, audio clip, video clip, or other work of media within that work itself. Although such practices have existed for quite a long time—at least several centuries, if not millennia—the field of *digital* watermarking only gained widespread popularity as a research topic in the latter half of the 1990s. A few earlier books have devoted substantial space to the subject of digital watermarking [171, 207, 219]. However, to our knowledge, this is the first book dealing exclusively with this field.

PURPOSE

Our goal with this book is to provide a framework in which to conduct research and development of watermarking technology. This book is not intended as a comprehensive survey of the field of watermarking. Rather, it represents our own point of view on the subject. Although we analyze specific examples from the literature, we do so only to the extent that they highlight particular concepts being discussed. (Thus, omissions from the Bibliography should not be considered as reflections on the quality of the omitted works.)

Most of the literature on digital watermarking deals with its application to images, audio, and video, and these application areas have developed somewhat independently. This is in part because each medium has unique characteristics, and researchers seldom have expertise in all three. We are no exception, our own backgrounds being predominantly in images and video. Nevertheless, the fundamental principles behind still image, audio, and video watermarking are the same, so we have made an effort to keep our discussion of these principles generic.

The principles of watermarking we discuss are illustrated with several example algorithms and experiments (the C source code is provided in Appendix C). All of these examples are implemented for image watermarking only. We decided to use only image-based examples because, unlike audio or video, images can be easily presented in a book.

The example algorithms are very simple. In general, they are not themselves useful for real watermarking applications. Rather, each algorithm is intended to provide a clear illustration of a specific idea, and the experiments are intended to examine the idea's effect on performance.

The book contains a certain amount of repetition. This was a conscious decision, because we assume that many, if not most, readers will not read the book from cover to cover. Rather, we anticipate that readers will look up topics of interest and read only individual sections or chapters. Thus, if a point is relevant in a number of places, we may briefly repeat it several times. It is hoped that this will not make the book too tedious to read straight through, yet will make it more useful to those who read technical books the way we do.

CONTENT AND ORGANIZATION

Chapters 1 and 2 of this book provide introductory material. Chapter 1 provides a history of watermarking, as well as a discussion of the characteristics that distinguish watermarking from the related fields of data hiding and steganography. Chapter 2 describes a wide variety of applications of digital watermarking and serves as motivation. The applications highlight a variety of sometimes conflicting requirements for watermarking, which are discussed in more detail in the second half of the chapter.

The technical content of this book begins with Chapter 3, which presents several frameworks for modeling watermarking systems. Along the way, we describe, test, and analyze some simple image watermarking algorithms that illustrate the concepts being discussed. In Chapter 4, these algorithms are extended to carry larger data payloads by means of conventional message-coding techniques. Although these techniques are commonly used in watermarking systems, some recent research suggests that substantially better performance can be achieved by exploiting side information in the encoding process. This is discussed in Chapter 5.

Chapter 7 analyzes message errors, false positives, and false negatives that may occur in watermarking systems. It also introduces whitening.

The next three chapters explore a number of general problems related to fidelity, robustness, and security that arise in designing watermarking systems, and present techniques that can be used to overcome them. Chapter 8 examines the problems of modeling human perception, and of using those models in watermarking systems. Although simple perceptual models for audio and still images are described, perceptual modeling is not the focus of this chapter. Rather, we focus on how any perceptual model can be used to improve the fidelity of the watermarked content.

Chapter 9 covers techniques for making watermarks survive several types of common degradations, such as filtering, geometric or temporal transformations, and lossy compression.

Chapter 10 describes a framework for analyzing security issues in watermarking systems. It then presents a few types of malicious attacks to which watermarks might be subjected, along with possible countermeasures.

Finally, Chapter 11 covers techniques for using watermarks to verify the integrity of the content in which they are embedded. This includes the area of fragile watermarks, which disappear or become invalid if the watermarked Work is degraded in any way.

ACKNOWLEDGMENTS

First, we must thank several people who have directly helped us in making this book. Thanks to Karyn Johnson, Jennifer Mann, and Marnie Boyd of Morgan Kaufmann for their enthusiasm and help with this book. As reviewers, Ton Kalker, Rade Petrovic, Steve Decker, Adnan Alattar, Aaron Birenboim, and Gary Hartwick provided valuable feedback. Harold Stone and Steve Weinstein of NEC also gave us many hours of valuable discussion. And much of our thinking about authentication (Chapter 11) was shaped by a conversation with Dr. Richard Green of the Metropolitan Police Service, Scotland Yard. We also thank M. Gwenael Doerr for his review.

Special thanks, too, to Valerie Tucci, our librarian at NEC, who was invaluable in obtaining many, sometimes obscure, publications. And Karen Hahn for secretarial support. Finally, thanks to Dave Waltz, Mitsuhito Sakaguchi, and NEC Research Institute for providing the resources needed to write this book. It could not have been written otherwise.

We are also grateful to many researchers and engineers who have helped develop our understanding of this field over the last several years. Our work on watermarking began in 1995 thanks to a talk Larry O'Gorman presented at NECI. Joe Kilian, Tom Leighton, and Talal Shamoon were early collaborators. Joe has continued to provide valuable insights and support. Warren Smith has taught us much about high-dimensional geometry. Jont Allen, Jim Flanagan, and Jim Johnston helped us understand auditory perceptual modeling. Thanks also to those at NEC Central Research Labs who worked with us on several watermarking projects: Ryoma Oami, Takahiro Kimoto, Atsushi Murashima, and Naoki Shibata.

Each summer we had the good fortune to have excellent summer students who helped solve some difficult problems. Thanks to Andy McKellips and Min Wu of Princeton University and Ching-Yung Lin of Columbia University. We also had the good fortune to collaborate with professors Mike Orchard and Stu Schwartz of Princeton University.

We probably learned more about watermarking during our involvment in the request for proposals for watermarking technologies for DVD disks than at any other time. We are therefore grateful to our competitors for pushing us to our limits, especially Jean-Paul Linnartz, Ton Kalker (again), and Maurice Maes of Philips; Jeffrey Rhoads of Digimarc; John Ryan and Patrice Capitant of Macrovision; and Akio Koide, N. Morimoto, Shu Shimizu, Kohichi Kamijoh, and Tadashi Mizutani of IBM (with whom we later collaborated). We are also grateful to the engineers of NEC's PC&C division who worked on hardware implementations for this competition, especially Kazuyoshi Tanaka, Junya Watanabe, Yutaka Wakasu, and Shigeyuki Kurahashi.

Much of our work was conducted while we were employed at Signafy, and we are grateful to several Signafy personnel who helped with the technical challenges: Peter Blicher, Yui Man Lui, Doug Rayner, Jan Edler, and Alan Stein (whose real-time video library is amazing).

We wish also to thank the many others who have helped us out in a variety of ways. A special thanks to Phil Feig—our favorite patent attorney—for filing many of our patent applications with the minimum of overhead. Thanks to Takao Nishitani for supporting our cooperation with NEC's Central Research Labs. Thanks to Kasinath Anupindi, Kelly Feng, and Sanjay Palnitkar for system administration support. Thanks to Jim Philbin, Doug Bercow, Marc Triaureau, Gail Berreitter, and John Anello for making Signafy a fun and functioning place to work. Thanks to Alan Bell for making CPTWG possible. Thanks to Mitsuhito Sakaguchi (again), who first suggested that we become involved in the CPTWG meetings. Thanks to Shichiro Tsuruta for managing PC&C's effort during the CPTWG competition, and H. Morito of NEC's semiconductor division. Thanks to Dan Sullivan for the part he played in our collaboration with IBM. Thanks to the DHSG cochairs who organized the competition: Bob Finger, Jerry Pierce, and Paul Wehrenberg. Thanks also to the many people at the Hollywood studios who provided us with the content owners' perspective: Chris Cookson and Paul Klamer of Warner Brothers, Bob Lambert of Disney, Paul Heimbach and Gary Hartwick of Viacom, Jane Sunderland and David Grant of Fox, David Stebbings of the RIAA, and Paul Egge of the MPAA. Thanks to Christine Podilchuk for her support. It was much appreciated. Thanks to Bill Connolly for interesting discussions. Thanks to John Kulp, Rafael Alonso, the Sarnoff Corporation, and John Manville of Lehman Brothers for their support. And thanks to Vince Gentile, Tom Belton, Susan Kleiner, Ginger Mosier, Tom Nagle, and Cynthia Thorpe.

Finally, we thank our families for their patience and support during this project: Susan and Zoe Cox, Geidre Miller, and Pamela Bloom.

Preface to the Second Edition

It has been almost 7 years since the publication of *Digital Watermarking*. During this period there has been significant progress in digital watermarking; and the field of steganography has witnessed increasing interest since the terrorist events of September 11, 2001.

Digital watermarking and steganography are closely related. In the first edition of *Digital Watermarking* we made a decision to distinguish between watermarking and steganography and to focus exclusively on the former. For this second edition we decided to broaden the coverage to include steganography and to therefore change the title of the book to *Digital Watermarking and Steganography*.

Despite the new title, this is *not* a new book, but a revision of the original. We hope this is clear from the backcover material and apologize in advance to any reader who thought otherwise.

CONTENT AND ORGANIZATION

The organization of this book closely follows that of the original. The treatment of watermarking and steganography is, for the most part, kept separate. The reasons for this are twofold. First, we anticipate that readers might prefer not to read the book from cover to cover, but rather read specific chapters of interest. And second, an integrated revision would require considerably more work.

Chapters 1 and 2 include new material related to steganography and, where necessary, updated material related to watermarking. In particular, Chapter 2 highlights the similarities and differences between watermarking and steganography.

Chapters 3, 4, 7, 8, 9, and 10 remain untouched, except that bibliographic citations have been updated.

Chapter 5 of the first edition has now been expanded to two chapters, reflecting the research interest in modeling watermarking as communications with side information. Chapter 5 provides a more detailed theoretical discussion of the topic, especially with regard to dirty-paper coding. Chapter 6 then provides a description of a variety of common dirty-paper coding techniques for digital watermarking.

Section 11.1.3 in Chapter 11 has been revised to include material on a variety of erasable watermarking methods.

Finally, two new chapters, Chapters 12 and 13, have been added. These chapters discuss steganography and steganalysis, respectively.

ACKNOWLEDGMENTS

The authors would like to thank the following people: Alan Bell of Warner Brothers for discussions on HD-DVD digital rights management technology, John Choi for discussions relating to watermarking of MP3 files in Korea, David Soukal for creating graphics for the Stego chapter.

And of course we would like to thank our families and friends for their support in the endeavor: Rimante Okkels; Zoe, Geoff, and Astrid Cox; Pam Bloom and her watermarking team of Joshua, Madison, Emily Giedre, Fia, and Ada; Monika, Nicole, and Kathy Fridrich; Miroslav Goljan; Robin Redding; and all the animals.

Finally, to Matt, your coauthors send their strongest wishes—get well soon!

Example Watermarking Systems

In this book, we present a number of example watermarking systems to illustrate and test some of the main points. Discussions of test results provide additional insights and lead to subsequent sections.

Each investigation begins with a preamble. If a new watermarking system is being used, a description of the system is provided. Experimental procedures and results are then described.

The watermark embedders and watermark detectors that make up these systems are given names and are referred to many times throughout the book. The naming convention we use is as follows: All embedder and detector names are written in sans serif font to help set them apart from the other text. Embedder names all start with E_ and are followed by a word or acronym describing one of the main techniques illustrated by an algorithm. Similarly, detector names begin with D_ followed by a word or acronym. For example, the embedder in the first system is named E_BLIND (it is an implementation of blind embedding), and the detector is named D_LC (it is an implementation of linear correlation detection).

Each system used in an investigation consists of an embedder and a detector. In many cases, one or the other of these is shared with several other systems. For example, in Chapter 3, the D_LC detector is paired with the E_BLIND embedder in System 1 and with the E_FIXED_LC embedder in System 2. In subsequent chapters, this same detector appears again in a number of other systems. Each individual embedder and detector is described in detail in the first system in which it is used.

In the following, we list each of the 19 systems described in the text, along with the number of the page on which its description begins, as well as a brief review of the points it is meant to illustrate and how it works. The source code for these systems is provided in Appendix C.

System 1: E_BLIND/D_LC 70

Blind Embedding and Linear Correlation Detection: The blind embedder E_BLIND simply adds a pattern to an image. A reference pattern is scaled by a strength parameter, α, prior to being added to the image. Its sign is dictated by the message being encoded.

The D_LC linear correlation detector calculates the correlation between the received image and the reference pattern. If the magnitude of the correlation is higher than a threshold, the watermark is declared to be present. The message is encoded in the sign of the correlation.

System 2: E_FIXED_LC/D_LC 77
Fixed Linear Correlation Embedder and Linear Correlation Detection: This system uses the same D_LC linear correlation detector as System 1, but introduces a new embedding algorithm that implements a type of informed embedding. Interpreting the cover Work as channel noise that is known, the E_FIXED_LC embedder adjusts the strength of the watermark to compensate for this noise, to ensure that the watermarked Work has a specified linear correlation with the reference pattern.

System 3: E_BLK_BLIND/D_BLK_CC 89
Block-Based, Blind Embedding, and Correlation Coefficient Detection: This system illustrates the division of watermarking into *media space* and *marking space* by use of an extraction function. It also introduces the use of the correlation coefficient as a detection measure.

The E_BLK_BLIND embedder performs three basic steps. First, a 64-dimensional vector, v_o, is extracted from the unwatermarked image by averaging 8 × 8 blocks. Second, a reference mark, w_r, is scaled and either added to or subtracted from v_o. This yields a marked vector, v_w. Finally, the difference between v_o and v_w is added to each block in the image, thus ensuring that the extraction process (block averaging), when applied to the resulting image, will yield v_w.

The D_BLK_CC detector extracts a vector from an image by averaging 8 × 8 pixel blocks. It then compares the resulting 64-dimensional vector, v, against a reference mark using the correlation coefficient.

System 4: E_SIMPLE_8/D_SIMPLE_8 116
8-Bit Blind Embedder, 8-Bit Detector: The E_SIMPLE_8 embedder is a version of the E_BLIND embedder modified to embed 8-bit messages. It first constructs a message pattern by adding or subtracting each of eight reference patterns. Each reference pattern denotes 1 bit, and the sign of the bit determines whether it is added or subtracted. It then multiplies the message pattern by a scaling factor and adds it to the image.

The D_SIMPLE_BITS detector correlates the received image against each of the eight reference patterns and uses the sign of each correlation to determine the most likely value for the corresponding bit. This yields the decoded message. The detector does not distinguish between marked and unwatermarked images.

System 5: E_TRELLIS_8/D_TRELLIS_8 123
Trellis-Coding Embedder, Viterbi Detector: This system embeds 8-bit messages using trellis-coded modulation. In the E_TRELLIS_8 embedder, the 8-bit

message is redundantly encoded as a sequence of symbols drawn from an alphabet of 16 symbols. A message pattern is then constructed by adding together reference patterns representing the symbols in the sequence. The pattern is then embedded with blind embedding.

The D_TRELLIS_8 detector uses a Viterbi decoder to determine the most likely 8-bit message. It does not distinguish between watermarked and unwatermarked images.

System 6: E_BLK_8/D_BLK_8 131

Block-Based Trellis-Coding Embedder and Block-Based Viterbi Detector That Detects by Reencoding: This system illustrates a method of testing for the presence of multibit watermarks using the correlation coefficient. The E_BLK_8 embedder is similar to the E_TRELLIS_8 embedder, in that it encodes an 8-bit message with trellis-coded modulation. However, it constructs an 8 × 8 message mark, which is embedded into the 8 × 8 average of blocks in the image, in the same way as the E_BLK_BLIND embedder.

The D_BLK_8 detector averages 8 × 8 blocks and uses a Viterbi decoder to identify the most likely 8-bit message. It then reencodes that 8-bit message to find the most likely message mark, and tests for that message mark using the correlation coefficient.

System 7: E_BLK_FIXED_CC/D_BLK_CC 144

Block-Based Watermarks with Fixed Normalized Correlation Embedding: This is a first attempt at informed embedding for normalized correlation detection. Like the E_FIXED_LC embedder, the E_BLK_FIXED_CC embedder aims to ensure a specified detection value. However, experiments with this system show that its robustness is not as high as might be hoped.

The E_BLK_FIXED_CC embedder is based on the E_BLK_BLIND embedder, performing the same basic three steps of extracting a vector from the unwatermarked image, modifying that vector to embed the mark, and then modifying the image so that it will yield the new extracted vector. However, rather than modify the extracted vector by blindly adding or subtracting a reference mark, the E_BLK_FIXED_CC embedder finds the closest point in 64 space that will yield a specified correlation coefficient with the reference mark. The D_BLK_CC detector used here is the same as in the E_BLK_BLIND/D_BLK_CC system.

System 8: E_BLK_FIXED_R/D_BLK_CC 149

Block-Based Watermarks with Fixed Robustness Embedding: This system fixes the difficulty with the E_BLK_FIXED_CC/D_BLK_CC system by trying to obtain a fixed estimate of robustness, rather than a fixed detection value.

After extracting a vector from the unwatermarked image, the E_BLK_FIXED_R embedder finds the closest point in 64 space that is likely to lie within the detection region even after a specified amount of noise has been added. The D_BLK_CC detector used here is the same as in the E_BLK_BLIND/D_BLK_CC system.

System 9: E_LATTICE/D_LATTICE 191

Lattice-Coded Watermarks: This illustrates a method of watermarking with dirty-paper codes that can yield much higher data payloads than are practical with the E_DIRTY_PAPER/D_DIRTY_PAPER system. Here, the set of code vectors is not random. Rather, each code vector is a point on a lattice. Each message is represented by all points on a sublattice.

The embedder takes a 345-bit message and applies an error correction code to obtain a sequence of 1,380 bits. It then identifies the sublattice that corresponds to this sequence of bits and quantizes the cover image to find the closest point in that sublattice. Finally, it modifies the image to obtain a watermarked image close to this lattice point.

The detector quantizes its input image to obtain the closest point on the entire lattice. It then identifies the sublattice that contains this point, which corresponds to a sequence of 1,380 bits. Finally, it decodes this bit sequence to obtain a 345-bit message. It makes no attempt to determine whether or not a watermark is present, but simply returns a random message when presented with an unwatermarked image.

System 10: E_E_8LATTICE/D_E_8LATTICE 202

E_8 Lattice-Coded Watermarks: This System illustrates the benefits of using an E_8 lattice over an orthogonal lattice, used in System 9. Experimental results compare the performance of System 10 and System 9 and demonstrate that the E_8 lattice has superior performance.

System 11: E_BLIND/D_WHITE 234

Blind Embedding and Whitened Linear Correlation Detection: This system explores the effects of applying a whitening filter in linear correlation detection. It uses the E_BLIND embedding algorithm introduced in System 1.

The D_WHITE detector applies a whitening filter to the image and the watermark reference pattern before computing the linear correlation between them. The whitening filter is an 11 × 11 kernel derived from a simple model of the distribution of unwatermarked images as an elliptical Gaussian.

System 12: E_BLK_BLIND/D_WHITE_BLK_CC 247
Block-Based Blind Embedding and Whitened Correlation Coefficient Detection: This system explores the effects of whitening on correlation coefficient detection. It uses the E_BLK_BLIND embedding algorithm introduced in System 3.

The D_WHITE_BLK_CC detector first extracts a 64 vector from the image by averaging 8×8 blocks. It then filters the result with the same whitening filter used in D_WHITE. This is roughly equivalent to filtering the image before extracting the vector. Finally, it computes the correlation coefficient between the filtered, extracted vector and a filtered version of a reference mark.

System 13: E_PERC_GSCALE 277
Perceptually Limited Embedding and Linear Correlation Detection: This system begins an exploration of the use of perceptual models in watermark embedding. It uses the D_LC detector introduced in System 1.

The E_PERC_GSCALE embedder is similar to the E_BLIND embedder in that, ultimately, it scales the reference mark and adds it to the image. However, in E_PERC_GSCALE the scaling is automatically chosen to obtain a specified perceptual distance, as measured by Watson's perceptual model.

System 14: E_PERC_SHAPE 284
Perceptually Shaped Embedding and Linear Correlation Detection: This system is similar to System 11, but before computing the scaling factor for the entire reference pattern the E_PERC_SHAPE embedder first *perceptually shapes* the pattern.

The perceptual shaping is performed in three steps. First, the embedder converts the reference pattern into the block DCT domain (the domain in which Watson's model is defined). Next, it scales each term of the transformed reference pattern by a corresponding *slack* value obtained by applying Watson's model to the cover image. This amplifies the pattern in areas where the image can easily hide noise, and attenuates in areas where noise would be visible. Finally, the resultant shaped pattern is converted back into the spatial domain. The shaped pattern is then scaled and added to the image in the same manner as in E_PERC_GSCALE.

System 15: E_PERC_OPT 290
Optimally Scaled Embedding and Linear Correlation Detection: This system is essentially the same as System 12. The only difference is that perceptual shaping is performed using an "optimal" algorithm, instead of simply scaling each term of the reference pattern's block DCT. This shaping is optimal in the sense

that the resulting pattern yields the highest possible correlation with the reference pattern for a given perceptual distance (as measured by Watson's model).

System 16: E_MOD/D_LC 381

Watermark Embedding Using Modulo Addition: This is a simple example of a system that produces erasable watermarks. It uses the D_LC detector introduced in System 1.

The E_MOD embedder is essentially the same as the E_BLIND embedder, in that it scales a reference pattern and adds it to the image. The difference is that the E_MOD embedder uses modulo 256 addition. This means that rather than being clipped to a range of 0 to 255, the pixel values wrap around. Therefore, for example, 253 + 4 becomes 1. Because of this wraparound, it is possible for someone who knows the watermark pattern and embedding strength to perfectly invert the embedding process, erasing the watermark and obtaining a bit-for-bit copy of the original.

System 17: E_DCTQ/D_DCTQ 400

Semi-fragile Watermarking: This system illustrates a carefully targeted semi-fragile watermark intended for authenticating images. The watermarks are designed to be robust against JPEG compression down to a specified quality factor, but fragile against most other processes (including more severe JPEG compression).

The E_DCTQ embedder first converts the image into the block DCT domain used by JPEG. It then quantizes several high-frequency coefficients in each block to either an even or odd multiple of a quantization step size. Each quantized coefficient encodes either a 0, if it is quantized to an even multiple, or a 1, if quantized to an odd multiple. The pattern of 1s and 0s embedded depends on a key that is shared with the detector. The quantization step sizes are chosen according to the expected effect of JPEG compression at the worst quality factor the watermark should survive.

The D_DCTQ detector converts the image into the block DCT domain and identifies the closest quantization multiples for each of the high-frequency coefficients used during embedding. From these, it obtains a pattern of bits, which it compares against the pattern embedded. If enough bits match, the detector declares that the watermark is present.

The D_DCTQ detector can be modified to yield localized information about where an image has been corrupted. This is done by checking the number of correct bits in each block independently. Any block with enough correctly embedded bits is deemed authentic.

System 18: E_SFSIG/D_SFSIG 406
Semi-fragile Signature: This extends the E_DCTQ/D_DCTQ system to provide detection of distortions that only effect the low-frequency terms of the block DCT. Here, the embedded bit pattern is a semi-fragile signature derived from the low-frequency terms of the block DCT.

The E_SFSIG embedder computes a bit pattern by comparing the magnitudes of corresponding low-frequency coefficients in randomly selected pairs of blocks. Because quantization usually does not affect the relative magnitudes of different values, most bits of this signature should be unaffected by JPEG (which quantizes images in the block DCT domain). The signature is embedded in the high-frequency coefficients of the blocks using the same method used in E_DCTQ.

The D_SFSIG detector computes a signature in the same way as E_SFSIG and compares it against the watermark found in the high-frequency coefficients. If enough bits match, the watermark is deemed present.

System 19: E_PXL/D_PXL 412
Pixel-by-Pixel Localized Authentication: This system illustrates a method of authenticating images with pixel-by-pixel localization. That is, the detector determines whether each individual pixel is authentic.

The E_PXL embedder embeds a predefined binary pattern, usually a tiled logo that can be easily recognized by human observers. Each bit is embedded in one pixel according to a secret mapping of pixel values into bit values (known to both embedder and detector). The pixel is moved to the closest value that maps to the desired bit value. Error diffusion is used to minimize the perceptual impact.

The D_PXL detector simply maps each pixel value to a bit value according to the secret mapping. Regions of the image modified since the watermark was embedded result in essentially random bit patterns, whereas unmodified regions result in the embedded pattern. By examining the detected bit pattern, it is easy to see where the image has been modified.

System 20: SE_LTSOLVER 463
Linear System Solver for Matrices Satisfying Robust Soliton Distribution: This system describes a method for solving a system of linear equations, $Ax = y$, when the Hamming weights of the matrix A columns follow a robust soliton distribution. It is intended to be used as part of a practical implementation of wet paper codes with non-shared selection rules.

The SE_LTSOLVER accepts on its input the linear system matrix, A, and the right hand side, y, and outputs the solution to the system if it exists,

or a message that the solution cannot be found. The solution proceeds by repeatedly swapping the rows and columns of the matrix until an upper diagonal matrix is obtained (if the system has a solution). The solution is then found by backsubstitution as in classical Gaussian elimination and re-permuting the solution vector.

System 21: SD_SPA 484

Detector of LSB Embedding: This is a steganalysis system that detects images with messages embedded using LSB embedding. It uses sample pairs analysis to estimate the number of flipped LSBs in an image and thereby detect LSB steganography.

It works by first dividing all pixels in the image into pairs and then assigns them to several categories. The cardinalities of the categories are used to form a quadratic equation for the unknown relative number of flipped LSBs. The input is a grayscale image, the output is the estimate of the relative message length in bits per pixel.

System 22: SD_DEN_FEATURES 491

Blind Steganalysis in Spatial Domain based on de-noising and a feature vector: This system extracts 27 features from a grayscale image for the purpose of blind steganlysis primarily in the spatial domain.

The SD_DEN_FEATURES system first applies a denoising filter to the image and then extracts the noise residual, which is subsequently transformed to the wavelet domain. Statistical moments of the coefficients from the three highest-frequency subbands are then calculated as features for steganalysis. Classification can be performed using a variety of machine learning tools.

CHAPTER 1
Introduction

Hold an American $20 bill up to the light. If you are looking at the side with the portrait of president Andrew Jackson, you will see that the portrait is echoed in a watermark on the right. This watermark is embedded directly into the paper during the papermaking process, and is therefore very difficult to forge. It also thwarts a common method of counterfeiting in which the counterfeiter washes the ink out of $20 bills and prints $100 bills on the same paper.

The watermark on the $20 bill (Figure 1.1), just like most paper watermarks today, has two properties that relate to the subject of the present book. First, the watermark is hidden from view during normal use, only becoming visible as a result of a special viewing process (in this case, holding the bill up to the light). Second, the watermark carries information about the object in which it is hidden (in this case, the watermark indicates the authenticity of the bill).

In addition to paper, watermarking can be applied to other physical objects and to electronic signals. Fabrics, garment labels, and product packaging are examples of physical objects that can be watermarked using special invisible dyes and inks [344, 348]. Electronic representations of music, photographs, and video are common types of signals that can be watermarked.

Consider another example that also involves imperceptible marking of paper but is fundamentally different on a philosophical level. Imagine a spy called Alice who needs to communicate a very important finding to her superiors. Alice begins by writing a letter describing her wonderful recent family vacation. After writing the letter, Alice replaces the ink in her pen with milk and writes a top secret message between the inked lines of her letter. When the milk dries, this secret message becomes imperceptible to the human eye. Heating up the paper above a candle will make the secret message visible. This is an example of steganography. In contrast to watermarking, the hidden message is unrelated to the content of the letter, which only serves as a decoy or cover to hide the very presence of sending the secret message.

FIGURE 1.1

American $20 bill.

This book deals with both watermarking and steganology[1] of electronic signals. We adopt the following terminology to describe these signals. We refer to a specific song, video, or picture—or to a specific copy of such—as a *Work,*[2] and to the set of all possible Works as *content*. Thus, audio music is an example of content, and the song "Satisfaction" by the Rolling Stones is an example of a Work. The original unaltered Work is sometimes referred to as the *cover Work,* in that it hides or "covers" the watermark or the secret message. We use the term *media* to refer to the means of representing, transmitting, and recording content. Thus, the audio CD on which "Satisfaction" is recorded is an example of a medium.

> *We define* watermarking *as the practice of imperceptibly altering a Work to embed a message about that Work.*[3]
> *We define* steganography *as the practice of undetectably altering a Work to embed a secret message.*

Even though the objectives of watermarking and steganography are quite different, both applications share certain high-level elements. Both systems

[1] We use the term *steganology* to refer to both steganography and steganalysis, just as cryptology refers to both cryptography and cryptanalysis. The term *steganology* is not commonly used but is more precise than using steganography. However, we will often use steganography and steganology interchangeably.

[2] This definition of the term *Work* is consistent with the language used in the United States copyright laws [416]. Other terms that have been used can be found in the disscussion of this term in the Glossary.

[3] Some researchers do not consider imperceptibility a defining characteristic of digital watermarking. This leads to the field of perceptible watermarking [52, 164, 286, 294, 295], which is outside the scope of this book.

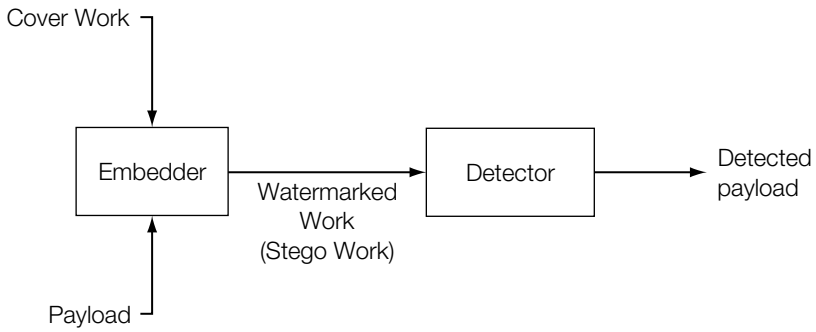


FIGURE 1.2

A generic watermarking (steganography) system.

consist of an *embedder* and a *detector,* as illustrated in Figure 1.2. The embedder takes two inputs. One is the payload we want to embed (e.g., either the watermark or the secret message), and the other is the cover Work in which we want to embed the payload. The output of the embedder is typically transmitted or recorded. Later, that Work (or some other Work that has not been through the embedder) is presented as an input to the detector. Most detectors try to determine whether a payload is present, and if so, output the message encoded by it.

In the late 1990s there was an explosion of interest in digital systems for the watermarking of various content. The main focus has been on photographs, audio, and video, but other content—such as binary images [453], text [49, 50, 271], line drawings [380], three-dimensional models [36, 312, 462], animation parameters [177], executable code [385], and integrated circuits [215, 249]—have also been marked. The proposed applications of these methods are many and varied, and include identification of the copyright owner, indication to recording equipment that the marked content should not be recorded, verification that content has not been modified since the mark was embedded, and the monitoring of broadcast channels looking for marked content.

Interest in steganology increased significantly after the terrorist attacks on September 11, 2001, when it became clear that means for concealing the communication itself are likely to be used for criminal activities.[4] The first steganalytic methods focused on the most common type of hiding called Least Significant Bit embedding [142, 444] in bitmap and GIF images. Later, substantial effort has been directed to the most common image format—JPEG—[132, 144] and audio files [443]. Accurate methods for detecting hidden messages prompted further research in steganography for multimedia files [147, 442].

[4] Interestingly, *USA Today* reported on this possibility several months before the September 11, 2001 attacks [1]. However, there has been little evidence to substantiate these claims.

1.1 INFORMATION HIDING, STEGANOGRAPHY, AND WATERMARKING

Information hiding, steganography, and watermarking are three closely related fields that have a great deal of overlap and share many technical approaches. However, there are fundamental philosophical differences that affect the requirements, and thus the design, of a technical solution. In this section, we discuss these differences.

Information hiding (or *data hiding*) is a general term encompassing a wide range of problems beyond that of embedding messages in content. The term *hiding* here can refer to either making the information imperceptible (as in watermarking) or keeping the existence of the information secret. Some examples of research in this field can be found in the International Workshops on Information Hiding, which have included papers on such topics as maintaining anonymity while using a network [232] and keeping part of a database secret from unauthorized users [298].

The inventor of the word *steganography* is Trithemius, the author of the early publications on cryptography: Polygraphia and Steganographia. The technical term itself is derived from the Greek words *steganos,* which means "covered," and *graphia,* which means "writing." Steganography is the art of concealed communication. The very existence of a message is secret. Besides invisible ink, an oft-cited example of steganography is an ancient story from Herodotus [192], who tells of a slave sent by his master, Histiæus, to the Ionian city of Miletus with a secret message tattooed on his scalp. After tattooing, the slave grew his hair back in order to conceal the message. He then journeyed to Miletus and, upon arriving, shaved his head to reveal the message to the city's regent, Aristagoras. The message encouraged Aristagoras to start a revolt against the Persian king. In this scenario, the message is of primary value to Histiæus and the slave is simply the carrier of the message.

We can use this example to highlight the difference between steganography and watermarking. Imagine that the message on the slave's head read, "This slave belongs to Histiæus." In that this message refers to the slave (cover Work), this would meet our definition of a watermark. Maybe the only reason to conceal the message would be cosmetic. However, if someone else claimed possession of the slave, Histiæus could shave the slave's head and prove ownership. In this scenario, the slave (cover Work) is of primary value to Histiæus, and the message provides useful information about the cover Work.

Systems for inserting messages in Works can thus be divided into watermarking systems, in which the message is related to the cover Work, and nonwatermarking systems, in which the message is unrelated to the cover Work. They can also be independently divided into steganographic systems, in which the very existence of the message is kept secret, and nonsteganographic systems, in which the existence of the message need not be secret.

Table 1.1 Four categories of information hiding. Each category is described with an example in the text. (CW refers to *cover Work*.)

	CW Dependent Message	**CW Independent Message**
Existence Hidden	Covert Watermarking (Example 1)	Steganography (Covert Communication) (Example 2)
Existence Known	Overt Watermarking (Example 3)	Overt Embedded Communications (Example 4)

This results in four categories of information-hiding systems, which are summarized in Table 1.1. An example of each of the four categories helps to clarify their definitions.

1. In 1981, photographic reprints of confidential British cabinet documents were being printed in newspapers. Rumor has it that to determine the source of the leak, Margaret Thatcher arranged to distribute uniquely identifiable copies of documents to each of her ministers. Each copy had a different word spacing that was used to encode the identity of the recipient. In this way, the source of the leaks could be identified [20]. This is an example of covert watermarking. The watermarks encoded information related to the recipient of each copy of the documents, and were covert in that the ministers were kept unaware of their existence so that the source of the leak could be identified.

2. The possibility of steganographically embedded data unrelated to the cover Work (i.e., messages hidden in otherwise innocuous transmissions) has always been a concern to the military. Simmons provides a fascinating description of covert channels [376], in which he discusses the technical issues surrounding verification of the SALT-II treaty between the United States and the Soviet Union. The SALT-II treaty allowed both countries to have many missile silos but only a limited number of missiles. To verify compliance with the treaty, each country would install sensors, provided by the other country, in their silos. Each sensor would tell the other country whether or not its silo was occupied, but nothing else. The concern was that the respective countries might design the sensor to communicate additional information, such as the location of its silo, hidden inside the legitimate message.

3. An example of an overt watermark (i.e., the presence of the watermark is known) can be seen at the web site of the Hermitage Museum in

St. Petersburg, Russia.[5] The museum presents a large number of high-quality digital copies of its famous collection on its web site. Each image has been watermarked to identify the Hermitage as its owner, and a message at the bottom of each web page indicates this fact, along with the warning that the images may not be reproduced. Knowledge that an invisible watermark is embedded in each image helps deter piracy.

4. Overt, embedded communication refers to the known transmission of auxiliary, hidden information that is unrelated to the signal in which it is embedded. It was common practice in radio in the late 1940s to embed a time code in the broadcast at a specified frequency (800 Hz, for example) [342]. This time code was embedded at periodic intervals, say every 15 minutes. The code was inaudibly hidden in the broadcast, but it was not a watermark because the message (the current time) was unrelated to the content of the broadcast. Further, it was not an example of steganography because the presence of an embedded time code can only be useful if its existence is known.

By distinguishing between embedded data that relates to the cover Work and hidden data that does not, we can anticipate the different applications and requirements of the data-hiding method. However, the actual techniques may be very similar, or in some cases identical. Thus, although this book focuses on watermarking and steganographic techniques, most of these techniques are applicable to other areas of information hiding.

1.2 HISTORY OF WATERMARKING

Although the art of papermaking was invented in China over one thousand years earlier, paper watermarks did not appear until about 1282, in Italy.[6] The marks were made by adding thin wire patterns to the paper molds. The paper would be slightly thinner where the wire was and hence more transparent.

The meaning and purpose of the earliest watermarks are uncertain. They may have been used for practical functions such as identifying the molds on which sheets of papers were made, or as trademarks to identify the paper maker. On the other hand, they may have represented mystical signs, or might simply have served as decoration.

By the eighteenth century, watermarks on paper made in Europe and America had become more clearly utilitarian. They were used as trademarks,

[5] See *http://www.hermitagemuseum.org*.

[6] Much of our description of paper watermarking is obtained from Hunter [204].

to record the date the paper was manufactured, and to indicate the sizes of original sheets. It was also about this time that watermarks began to be used as anticounterfeiting measures on money and other documents.

The term *watermark* seems to have been coined near the end of the eighteenth century and may have been derived from the German term *wassermarke* [378] (though it could also be that the German word is derived from the English [204]). The term is actually a misnomer, in that water is not especially important in the creation of the mark. It was probably given because the marks resemble the effects of water on paper.

About the time the term *watermark* was coined, counterfeiters began developing methods of forging watermarks used to protect paper money. In 1779, *Gentleman's Magazine* [285] reported that a man named John Mathison

> *. . . had discovered a method of counterfeiting the water-mark of the bank paper, which was before thought the principal security against frauds. This discovery he made an offer to reveal, and of teaching the world the method of detecting the fraud, on condition of pardon, which, however, was no weight with the bank.*

John Mathison was hanged.

Counterfeiting prompted advances in watermarking technology. William Congreve, an Englishman, invented a technique for making color watermarks by inserting dyed material into the middle of the paper during papermaking. The resulting marks must have been extremely difficult to forge, because the Bank of England itself declined to use them on the grounds that they were too difficult to make. A more practical technology was invented by another Englishman, William Henry Smith. This replaced the fine wire patterns used to make earlier marks with a sort of shallow relief sculpture, pressed into the paper mold. The resulting variation on the surface of the mold produced beautiful watermarks with varying shades of gray. This is the basic technique used today for the face of President Jackson on the $20 bill.

Examples of our more general notion of watermarks—imperceptible messages about the objects in which they are embedded—probably date back to the earliest civilizations. David Kahn, in his classic book *The Codebreakers*, provides interesting historical notes [214]. An especially relevant story describes a message hidden in the book *Hypnerotomachia Poliphili,* anonymously published in 1499. The first letters of each chapter spell out "Poliam Frater Franciscus Columna Peramavit," assumed to mean "Father Francesco Columna loves Polia."[7]

---

[7] This translation is not universally accepted. Burke [58] notes that the two words of the title of the book are made up by the author. Burke goes on to claim that the word *Poliphili,* assumed to mean "lover of Polia," might also mean "the Antiquarian"; in which case, the secret message might be better translated as "Father Francesco Columna was deeply devoted to archeological studies."

Four hundred years later, we find the first example of a technology similar to the digital methods discussed in this book. In 1954, Emil Hembrooke of the Muzak Corporation filed a patent for "watermarking" musical Works. An identification code was inserted in music by intermittently applying a narrow notch filter centered at 1 kHz. The absence of energy at this frequency indicated that the notch filter had been applied and the duration of the absence used to code either a dot or a dash. The identification signal used Morse code. The 1961 U.S. Patent describing this invention states [185]:

> *The present invention makes possible the positive identification of the origin of a musical presentation and thereby constitutes an effective means of preventing such piracy, i.e. it may be likened to a watermark in paper.*

This system was used by Muzak until around 1984 [432]. It is interesting to speculate that this invention was misunderstood and became the source of persistent rumors that Muzak was delivering subliminal advertising messages to its listeners.

It is difficult to determine when *digital* watermarking was first discussed. In 1979, Szepanski [398] described a machine-detectable pattern that could be placed on documents for anti-counterfeiting purposes. Nine years later, Holt *et al*. [197] described a method for embedding an identification code in an audio signal. However, it was Komatsu and Tominaga [238], in 1988, who appear to have first used the term *digital watermark*. Still, it was probably not until the early 1990s that the term *digital watermarking* really came into vogue.

About 1995, interest in digital watermarking began to mushroom. Figure 1.3 is a histogram of the number of papers published on the topic [63]. The first Information Hiding Workshop (IHW) [20], which included digital watermarking as one of its primary topics, was held in 1996. The SPIE began devoting a conference specifically to *Security and Watermarking of Multimedia Contents* [450, 451], beginning in 1999.

In addition, about this time, several organizations began considering watermarking technology for inclusion in various standards. The Copy Protection Technical Working Group (CPTWG) [34] tested watermarking systems for protection of video on DVD disks. The Secure Digital Music Initiative (SDMI) [364] made watermarking a central component of their system for protecting music. Two projects sponsored by the European Union, VIVA [110] and Talisman [180], tested watermarking for broadcast monitoring. The International Organization for Standardization (ISO) took an interest in the technology in the context of designing advanced MPEG standards.

In the late 1990s several companies were established to market watermarking products. Technology from the Verance Corporation was adopted into the first phase of SDMI and was used by Internet music distributors such as Liquid Audio. In the area of image watermarking, Digimarc bundled its watermark embedders and detectors with Adobe's Photoshop. More recently, a number of

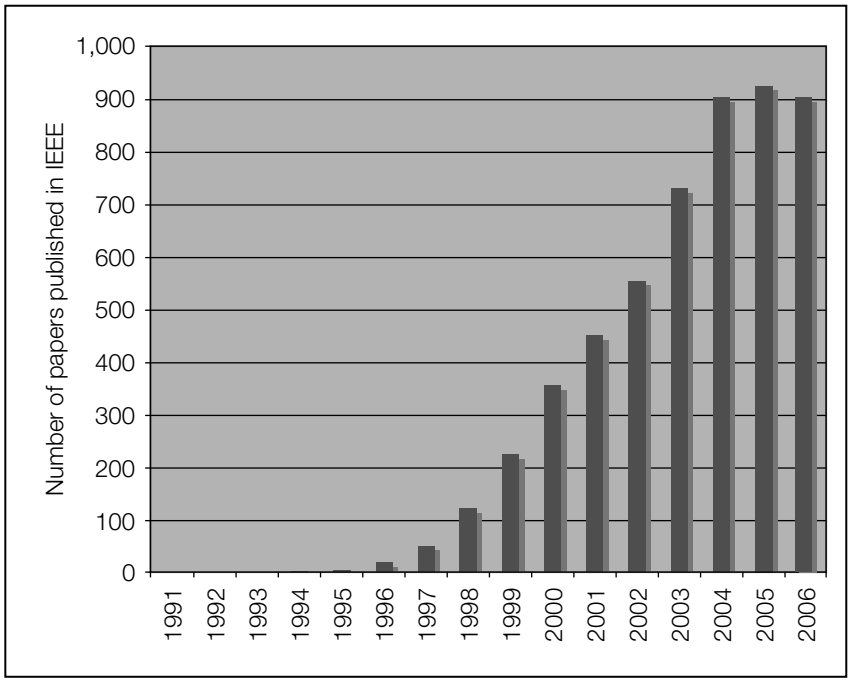


FIGURE 1.3

Annual number of papers published on watermarking and steganography by the IEEE.

companies have used watermarking technologies for a variety of applications, which are discussed in detail in Chapter 2.

1.3 HISTORY OF STEGANOGRAPHY

The first written evidence about steganography being used to send messages is the Herodotus [192] story about slaves and their shaved heads already mentioned in Section 1.1. Herodotus also documented the story of Demeratus, who alerted Sparta about the planned invasion of Greece by the Persian Great King Xerxes. Demeratus scraped the wax off the surface of a wooden writing tablet and scratched his warning into the wood. The tablet was then coated with a fresh layer of wax to appear as a blank writing tablet that could be safely carried to Sparta without arousing suspicion.

Aeneas the Tactician [399] proposed many steganographic techniques that could be considered "state of the art" of his time, such as hiding messages in women's earrings or messages carried by pigeons. He also described several methods for hiding in text—by modifying the height of letter strokes or marking letters in a text using small holes.

Linguistic steganography, also called *acrostic*, was one of the most popular ancient steganographic methods. Secret messages were encoded as initial letters of sentences or successive tercets in a poem. One of the most famous examples is *Amorosa visione* by Giovanni Boccacio [446].

A more advanced version of linguistic steganography originally conceived in China and reinvented by Cardan (1501–1576) is the famous Cardan's Grille. The letters of the secret message do not form a regular structure but a random pattern. The message is read simply by placing a mask over the text. The mask is an early example of a secret (stego) key that had to be shared between communicating parties. Acrostic was also used in World War I by both the Germans and Allies.

A precursor of modern steganographic methods was described by François Bacon [27]. Bacon used italic or normal fonts to encode binary representations of letters in his works. Five letters of the cover Work could hold five bits and thus one letter of the alphabet. What made this method relatively inconspicuous was the variability of sixteenth-century typography.

A modern version of this steganographic technique was described by Brassil *et al.* [51]. They used the fact that while shifting lines of text up or down by 1/300 of an inch is not visually perceptible, these small changes are robust enough to survive photocopying.

Another idea that played an important role in several wars in the nineteenth and twentieth centuries was originally proposed by Brewster (1857) [54]. He suggested hiding messages by shrinking them so much that they started resembling specs of dirt. The shrinking was made possible by the technology developed by French photographer Dragon during the Franco-Prussian War (1870–1871). Microscopic images could be hidden in nostrils, ears, or under fingernails [386]. In World War I, Germans used such "microdots" and hid them in corners of postcards slit open with a knife and resealed with starch. The modern twentieth-century microdots could hold up to one page of text and even contain photographs. The Allies discovered the use of microdots in 1941.

A more recent and quite ingenious use of steganography helped Commander Jeremiah Denton convey the truth about his North Vietnamese captors. When paraded in front of the news media as part of staged propaganda, Denton blinked his eyes in Morse code spelling out T-O-R-T-U-R-E.

Similar to watermarking, the boom of steganography coincides with the appearance of the Internet. The rapid spread of computer networks and shift to digitization of media created a very favorable environment for covert steganographic communication. Recently, steganography has been suspected as a possible means of information exchange and planning of terrorist attacks. It is only natural that such technology by its very nature could be used for planning criminal activities. Moreover, as of writing this book in mid-2006, there are over 300 steganographic products on the Internet available for download today. Some of these tools offer strong encryption methods that encrypt the secret messages

to provide an additional layer of security in case the steganographic scheme is broken. An example of such a program is Steganos (*http://www.steganos.com/*) or Stealthencrypt (*http://www.stealthencrypt.com/*). For more current lists of steganographic programs, see *http://www.stegoarchive.com/*.

Advances in steganography have spurred the complementary field of steganalysis that started developing more rapidly after the terrorist attacks of September 11, 2001. Steganalysis is concerned with developing methods for detecting the presence of secret messages and eventually extracting them. Steganography is considered broken even when the mere *presence* of the secret message is detected. Indeed, the fact that we know that certain parties are communicating secretly is often a very important piece of information.

1.4 IMPORTANCE OF DIGITAL WATERMARKING

The sudden increase in watermarking interest is most likely due to the increase in concern over copyright protection of content. The Internet had become user friendly with the introduction of Marc Andreessen's Mosaic web browser in November 1993 [11], and it quickly became clear that people wanted to download pictures, music, and videos. The Internet is an excellent distribution system for digital media because it is inexpensive, eliminates warehousing and stock, and delivery is almost instantaneous. However, content owners (especially large Hollywood studios and music labels) also see a high risk of piracy.

This risk of piracy is exacerbated by the proliferation of high-capacity digital recording devices. When the only way the average customer could record a song or a movie was on analog tape, pirated copies were usually of a lower quality than the originals, and the quality of second-generation pirated copies (i.e., copies of a copy) was generally very poor. However, with digital recording devices, songs and movies can be recorded with little, if any, degradation in quality. Using these recording devices and using the Internet for distribution, would-be pirates can easily record and distribute copyright-protected material without appropriate compensation being paid to the actual copyright owners. Thus, content owners are eagerly seeking technologies that promise to protect their rights.

The first technology content owners turn to is cryptography. Cryptography is probably the most common method of protecting digital content. It is certainly one of the best developed as a science. The content is encrypted prior to delivery, and a decryption key is provided only to those who have purchased legitimate copies of the content. The encrypted file can then be made available via the Internet, but would be useless to a pirate without an appropriate key. Unfortunately, encryption cannot help the seller monitor how a legitimate customer handles the content after decryption. A pirate can actually purchase the product, use the decryption key to obtain an unprotected copy of the

content, and then proceed to distribute illegal copies. In other words, cryptography can protect content in transit, but once decrypted, the content has no further protection.

Thus, there is a strong need for an alternative or complement to cryptography: a technology that can protect content even after it is decrypted. Watermarking has the potential to fulfill this need because it places information within the content where it is never removed during normal usage. Decryption, reencryption, compression, digital-to-analog conversion, and file format changes—a watermark can be designed to survive all of these processes.

Watermarking has been considered for many copy prevention and copyright protection applications. In copy prevention, the watermark may be used to inform software or hardware devices that copying should be restricted. In copyright protection applications, the watermark may be used to identify the copyright holder and ensure proper payment of royalties.

Although copy prevention and copyright protection have been major driving forces behind research in the watermarking field, there are a number of other applications for which watermarking has been used or suggested. These include broadcast monitoring, transaction tracking, authentication (with direct analogy to our $20 bill example), copy control, and device control. These applications are discussed in Chapter 2.

1.5 IMPORTANCE OF STEGANOGRAPHY

Electronic communication is increasingly susceptible to eavesdropping and malicious interventions. The issues of security and privacy have traditionally been approached using tools from cryptography. Messages can be appended with a message authentication code (hash) and encrypted so that only the rightful recipient can read them and verify their integrity and authenticity. Modern cryptography is a mature field based on rigorous mathematical foundations and decades of development.

Encrypted messages are obvious, and when intercepted, it is clear that the sender and the recipient are communicating secretly. Steganography is the little and much younger sister of cryptography. It is an alternative tool for privacy and security. Instead of encrypting messages, we can hide them in other innocuous-looking objects so that their very presence is not revealed. Thus, steganography can be a feasible alternative in countries where usage of encryption is illegal or in oppressive regimes where using cryptography might attract unwanted attention. A recent example of a practical steganographic scheme that was used for information exchange between two subjects, one of which was residing at the time in a hostile country, was described by Toby Sharp at the 4th Information Hiding Workshop [367].

As cryptanalysis is the other side of the cryptography coin, so steganalysis is an inseparable part of steganography. Indeed, one probably cannot develop a good steganographic method without spending a substantial amount of time on how to break it.

The need for reliable steganalytic tools capable of detecting hidden messages has recently increased due to anecdotal evidence that steganography is being used by terrorists and child pornographers. However, we have been unable to substantiate these claims. The closest support we found was from a *New York Times* article from November 6, 2006. An Al Qaeda operative, Dhiren Barot, filmed reconnaissance video between Broadway and South Street and concealed it before distribution by splicing it to the end of a copy of the Bruce Willis movie *Die Hard: With a Vengeance*. A primitive form of steganography.

This page intentionally left blank

CHAPTER 2

Applications and Properties

Watermarking can be used in a wide variety of applications. In general, if it is useful to associate some additional information with a Work, this metadata can be embedded as a watermark. Of course, there are other ways to associate information with a Work, such as placing it in the header of a digital file, encoding it in a visible bar code on an image, or speaking it aloud as an introduction to an audio clip. The questions arise: When is watermarking a better alternative? What can watermarking do that cannot be done with simpler techniques?

Watermarking is distinguished from other techniques in three important ways. First, watermarks are imperceptible. Unlike bar codes, they do not detract from the aesthetics of an image. Second, watermarks are inseparable from the Works in which they are embedded. Unlike header fields, they do not get removed when the Works are displayed or converted to other file formats. Finally, watermarks undergo the same transformations as the Works. This means that it is sometimes possible to learn something about those transformations by looking at the resulting watermarks. It is these three attributes that make watermarking invaluable for certain applications.

The performance of a given watermarking system can be evaluated on the basis of a small set of properties. For example, *robustness* describes how well watermarks survive common signal processing operations, *fidelity* describes how imperceptible the watermarks are, and so forth. The relative importance of these properties depends on the application for which the system is designed. For example, in applications where we have to detect the watermark in a copy of a Work that has been broadcast over an analog channel, the watermark must be robust against the degradation caused by that channel. However, if we can reasonably expect that a Work will not be modified at all between embedding and detection, the watermark's robustness is irrelevant.

Steganography is distinguished from other techniques in that it is covert (i.e., only the communicating parties, for example, Alice and Bob, are aware of the communication). While watermarking is also a form of data hiding, in many applications, the presence of watermarking is well known. And while encryption provides for privacy (i.e., a third party is not aware of *what* Alice

and Bob are communicating), encryption does not hide the fact that Alice and Bob *are* communicating. And it is this fact that Alice and Bob wish to conceal using steganography.

The performance of a given steganographic system can be evaluated on the basis of several properties. Most important, *statistical undetectability* describes how difficult it is to reliably determine the existence of a hidden message in a Work. A related measure is *steganographic capacity*, which is the maximum payload that can safely be hidden in a Work without causing statistically detectable artifacts. Different applications may require different payloads. For very short payloads, say 20 bits or less, it is almost impossible for an adversary to detect the presence of steganography. Steganographic research is therefore focused on the design of steganographic algorithms that permit long messages, perhaps thousands of bits, to be embedded in an undetectable manner.

In this chapter, we first describe several applications that can be implemented with watermarking and steganography and examine the advantages these approaches might have over alternative technologies. This discussion includes several examples of planned or actual watermarking systems in use today. We then describe several properties of watermarking and steganographic systems, discussing how their relative importance and interpretation varies with application. The corresponding sections end with a brief discussion of how performance can be assessed.

2.1 APPLICATIONS OF WATERMARKING

We examine eight proposed or actual watermarking applications: broadcast monitoring, owner identification, proof of ownership, transaction tracking, authentication, copy control, device control, and legacy enhancements. For each of these applications, we try to identify what characteristics of the problem make watermarking a suitable solution. To do this we must carefully consider the application requirements and examine the limitations of alternative solutions.

2.1.1 Broadcast Monitoring

In 1997, a scandal broke out in Japan regarding television advertising. At least two stations had been routinely overbooking air time. Advertisers were paying for thousands of commercials that were never aired [235]. The practice had remained largely undetected for more than 20 years, in part because there were no systems in place to monitor the actual broadcast of advertisements.

There are several types of organizations and individuals interested in broadcast monitoring. Advertisers, of course, want to ensure that they receive all of the air time they purchase from broadcasters, such as the Japanese television stations caught in the 1997 scandal. Performers, in turn, want to ensure that

they get the royalties due to them from advertising firms. A 1999 spot check by the Screen Actor's Guild (SAG) found an average of $1,000 in underpaid royalties per hour of U.S. television programming [356]. In addition, owners of copyrighted Works want to ensure that their property is not illegally rebroadcast by pirate stations.

A low-tech method of broadcast monitoring is to have human observers watch the broadcasts and record what they see or hear. This method is costly and error prone. It is therefore highly desirable to replace it with some form of automated monitoring. Techniques for doing this can be broadly broken down into two categories. *Passive* monitoring systems try to directly recognize the content being broadcast, in effect simulating human observers (though more reliably and at lower cost). *Active* monitoring systems rely on associated information that is broadcast along with the content.

A passive system consists of a computer that monitors broadcasts and compares the received signals with a database of known Works. When the comparison locates a match, the song, film, TV program, or commercial being aired can be identified. This is the most direct and least intrusive method of automated broadcast monitoring. It does not require the introduction of any associated information into the broadcast, and therefore does not require changes to advertisers' workflow. In fact, it does not require any cooperation with the advertisers or broadcasters.

However, there are a number of potential problems with implementing passive monitoring systems. First, comparing the received signals against a database is not trivial. In principle, we would like to divide the signals into recognizable units, such as individual frames of video, and search for them in the database. However, each frame of video consists of several million bits of information, and it would be impractical to use such a large bit sequence as an index for a database search. Thus, the system must first process the received signals into smaller signatures that are rich enough to differentiate between all possible Works yet small enough to be used as indices in a database search. Defining these signatures is difficult. Furthermore, broadcasting generally degrades the signals, and this degradation might vary over time, with the result that multiple receptions of the same Work at different times might lead to different signatures. This means that the monitoring system cannot search for an exact match in its database. Instead, it must perform a nearest-neighbor search, which is known to be substantially more complex [47, 69]. Because of the difficulty of deriving meaningful signatures and searching for nearest neighbors in a large database, it is difficult to design a passive monitoring system that is 100% reliable.

Even if the problem of searching the database is solved, storing and managing the database can be expensive because the database is large. Furthermore, the system should monitor several geographic locations simultaneously. In the United States, for example, we might want to monitor each of the roughly 120 television markets. Consequently, each site must either access a centralized

database of known Works, store the database locally (with a capability for regular updating), or transmit the signatures back to a central processing facility.

Is such a system cost effective for broadcast monitoring? Of course this depends on the value customers place on the information collected. In particular, because passive monitoring does not require the cooperation of those being monitored, it allows the monitoring service to tabulate competitive market research data. For example, such a system can be used to help Pepsi estimate how much Coca-Cola spends on advertising in the Atlanta market. Because of this, both Nielsen Media Research (NMR) and Competitive Media Reporting (CMR) have deployed passive systems for monitoring television broadcast.[1] However, neither company uses passive monitoring for verification purposes (i.e., to verify that a commercial was broadcast). The reason for this might be that the recognition system is simply not accurate enough. An accuracy of, say, 95% is adequate for acquiring competitive market research data. However, an error rate of 5% is too high for verification services. Imagine the consequences of erroneously informing Coca-Cola that 1 in 20 of their commercials did not air.

Despite these concerns, there has been recent progress in passive broadcast monitoring. Oostveen *et al.* [313] and others [18, 81, 135, 169, 170, 292, 293, 309] have reported very interesting results for audio fingerprinting. Oostveen *et al.*'s work has been commercialized by Philips Corporation and licenced to a number of companies including Nielsen Broadcast Data Systems (*www.bdsonline.com*), Gracenote (*www.gracenote.com*), Musiwave (*www.musiwave.net*), Optimus (*www.optimus.pt*), Amena (*www.amena.com*), and Snocap (*www.snocap.com*).

Obtaining the accuracy required for a verification service probably requires an active monitoring system, in which computer-recognizable identification information is transmitted along with content. Active monitoring is technically simpler to implement than passive monitoring. The identification information is straightforward to decode reliably, and no database is required to interpret its meaning.

One way to implement an active system is to place the identification information in a separate area of the broadcast signal. For example, analog television broadcasts permit digital information to be encoded in the vertical blanking interval (VBI) of a video signal. This part of the signal, sent between frames, has no effect on the picture. Closed captioning information is distributed in this manner, as is Teletext in Europe. Nielsen Media Research uses the VBI for its SIGMA advertisement monitoring service.

Unfortunately, embedding signals in the VBI can be problematic. In the United States, it is not always clear who legally controls the content of the VBI. Both cable and satellite operators stake a claim to it. Although U.S. law

[1] In fact, passive monitoring dates back at least to 1975 [296].

requires that the distributors deliver closed caption data to their customers in line 21 of the VBI [406, Sec. 305, 713] [405], distributors have no legal obligation to maintain any other information that may have been inserted into the VBI by content providers. Consequently, an identification code placed in the VBI by an advertiser or network is not guaranteed to be transmitted. Moreover, the appended information is unlikely to survive format changes, such as conversion from analog to digital, without modification of existing hardware. In the past, this was not a significant problem, but it is increasingly a concern as television broadcasts move from analog to digital transmission.

For digital Works, there are similar active techniques that store identification codes in file headers. These techniques suffer the same types of problems as the VBI approach, in that intermediate handlers and ultimate distributors would have to guarantee delivery of the header information intact. Again, the data is unlikely to survive format changes without modification to existing systems.

Watermarking is an obvious alternative method of coding identification information for active monitoring. It has the advantage of existing within the content itself, rather than exploiting a particular segment of the broadcast signal, and is therefore completely compatible with the installed base of broadcast equipment, including both digital and analog transmission. The primary disadvantage is that the embedding process is more complicated than placing data in the VBI or in file headers. There is also a concern, especially on the part of content creators, that the watermark may degrade the visual or audio quality of the Work. Nevertheless, there are a number of companies that provide watermark-based broadcast monitoring services. For example, Teletrax offers a service that is based on video watermarking technology from Philips.

2.1.2 Owner Identification

Under U.S. law, the creator of a story, painting, song, or any other original Work automatically holds copyright to it the instant the Work is recorded in some physical form.[2] Through 1988, if copyright holders wanted to distribute their Works without losing any rights, they had to include a copyright notice in every distributed copy. After 1988, this was changed so that the copyright notice is now no longer required. However, if a Work that is protected by copyright is misused, and the courts choose to award the copyright holder damages, that award can be significantly limited if a copyright notice of acceptable form and placement was not found on the distributed material [416, Sec. 401].

The exact form of the copyright notice is important. For visual Works, it must say either "Copyright *date owner,*" "© *date owner,*" or "Copr. *date owner*."

[2] Disclaimer: We are not lawyers. Although we have made some effort to ensure our information is correct, anything we say about American law or any other country's laws is interpretive.

Note that using "(C)" instead of "©" does not serve the same legal purpose. For sound recordings, the copyright notice takes the similar form "℗ *date owner*" and must be placed on the surface of the physical media, the label, or on the packaging so as to "give reasonable notice of the claim of copyright" [416, Sec. 402].

Textual copyright notices have several limitations as a technology for identifying the owner of a Work. For one, they are easy to remove from a document when it is copied, even without any intention of wrongdoing. For example, a professor copying pages out of a book (within the strictures of fair use) might neglect to photocopy the copyright notice on the title page. An artist using a legally acquired photograph in a magazine advertisement might crop off the portion of it that includes the copyright notice. Thus, a law-abiding citizen who subsequently wishes to to use a Work may not be able to determine whether the Work is protected by copyright. Even if the Work is assumed to be protected, it may be difficult to find the identity of the creator or person whose permission must be obtained.

A famous case in which the loss of text on an image caused just such problems is a photograph of Lena Sjööblom. This is perhaps the most common test image in image processing research and has appeared in countless journal articles and conference proceedings—all without any reference to its rightful owner, Playboy Enterprises, Inc. The image started out as a *Playboy* centerfold [334]. When this image was scanned for use as a test image, most of the image was cropped, leaving only Lena's face and shoulder. Unfortunately, in the process, the text that identified Playboy as the owner was also cropped (see Figure 2.1). The image has since been distributed electronically around the world, and most researchers who include it in their papers are probably unaware that they are infringing Playboy's copyright. Playboy has decided to overlook the widespread use of this image [56].

Another problem with textual copyright notices for images is that they can be aesthetically ugly and may cover a portion of the image. Although it is usually possible to make them unobtrusive [52] (e.g., by placing them in an unimportant corner of the picture), such practice makes them more susceptible to being cropped. The situation is even worse in audio, where the copyright notice is placed on the physical medium (disk, tape, record, and so on) and on the packaging. Neither of these notices would normally be copied along with the audio content. In fact, for some audio content that may exist only in electronic form—on a web site, for example—no physical medium or packaging would even exist.

Because watermarks can be made both imperceptible and inseparable from the Work that contains them, they are likely to be superior to text for owner identification. If users of Works are supplied with watermark detectors, they should be able to identify the owner of a watermarked Work, even after the Work has been modified in ways that would remove a textual copyright notice. Digimarc's watermark for images was designed with precisely this application

FIGURE 2.1

The often-used Lena image in image processing research is a cropped version of a 1972 *Playboy* centerfold. This portion of the original, lost to cropping, identifies the copyright owner.

in mind. They achieved widespread distribution of their watermark detector by bundling it with Adobe's popular image processing program, Photoshop. When Digimarc's detector recognizes a watermark, it contacts a central database over the Internet, and uses the watermark message as a key to find contact information for the image's owner.

The legal impact of such a system has not yet been tested in court. At present, given that the exact form of a copyright notice holds such legal significance, a copyright notice in a watermark probably would not suffice as an alternative to including the standard "©" notice. However, the system does make it easier for honest people to find out who they should contact about using an image.

2.1.3 **Proof of Ownership**

It is enticing to try to use watermarks not just to *identify* copyright ownership but to actually *prove* ownership. This is something a textual notice cannot do, because it can be so easily forged. For example, suppose an artist (call her

Alice) creates an image and posts it on her web site, with the copyright notice "© 2001 Alice." An adversary (call him Bob) then steals the image, uses an image processing program to replace the copyright notice with "© 2001 Bob," and then claims to own the copyright himself. How is the dispute resolved?

One way of resolving such a dispute is by use of a central repository. Before putting her image on the Web, Alice could register the image with the United States Copyright Office by sending a copy to them. They archive the image, together with information about the rightful owner. Then, when a dispute between Alice and Bob arises, Alice contacts the United States Copyright Office to prove that she is the rightful owner.

However, Alice might decline to register her image because it is too costly. Registering with the United States Copyright Office costs approximately $45 per document.[3] With many images to be registered, this can add up to a substantial expense for a struggling artist. If Alice cannot afford this expense, she might find herself prosecuting Bob without the benefit of the United States Copyright Office on her side.

In such a case, Alice would have to show evidence that she created the image. For example, she might have the film negative if the image was originally a photograph. Or she might have early drafts if it is a Work of art. The trouble is, if Bob is truly determined to win the case, he can fabricate such evidence himself. He can make a new negative from the image, or falsely manufacture early drafts of his own. Even worse, if the image was created digitally, there might not have been any negative or early drafts in the first place.

Can Alice protect her rights, and avoid incurring the cost of registration, by applying a watermark to her image? In the case of Digimarc's watermark, the answer is probably no. The problem with their system is that the detector is readily available to adversaries. As described in Section 2.3.7, and elaborated upon in Chapter 10, anybody who can detect a watermark can probably remove it. Thus, because Bob can easily obtain a Digimarc detector, he can remove Alice's watermark and replace it with his own.

To achieve the level of security required for proof of ownership, it is probably necessary to restrict the availability of the detector. When an adversary does not have a detector, removal of a watermark can be made extremely difficult. Therefore, when Alice and Bob go before the judge, Alice would produce her original copy of the image. Her original, and the disputed copy, would be entered into the watermark detector, and the detector would detect Alice's watermark.

However, even if Alice's watermark cannot be removed, Bob might be able to undermine her. As pointed out by Craver *et al.* [101], Bob, using his own

[3] The current registration fee can be obtained from the copyright office web site at *http://www.copyright.gov/docs/fees.html*. The $45 cited represents the advertised fee at the time of this writing.

watermarking system, might be able to make it appear as though *his* watermark were present in *Alice's* original copy of the image. Thus, a third party would be unable to judge whether Alice or Bob had the true original. This is described in more detail in Chapter 10.

This problem can be solved if we make a slight change in the problem statement. Instead of trying to directly prove ownership by embedding an "Alice owns this image" watermark message in it, we will instead try to prove that one image is derived from another. How this can be done is discussed in Chapter 10. Such a system provides indirect evidence that it is more likely the disputed image is owned by Alice than by Bob, in that Alice is the one who has the version from which the other two were created. The evidence is similar to that provided if Alice were to produce the negative from which the image was created, except that it is stronger, in that Bob can fabricate a negative but cannot fabricate a fake original that passes our test.

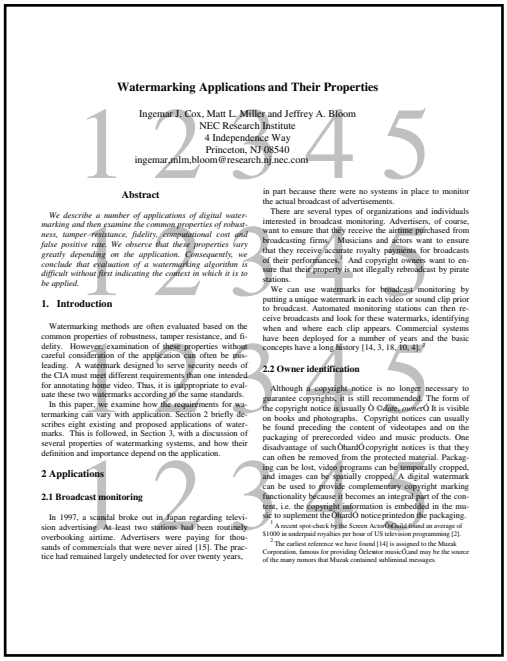
2.1.4 **Transaction Tracking**

In this application of watermarking, the watermark records one or more transactions that have taken place in the history of the copy of a Work in which it is embedded. For example, the watermark might record the recipient in each legal sale or distribution of the Work. The owner or producer of the Work would place a different watermark in each copy. If the Work were subsequently misused (leaked to the press or redistributed illegally), the owner could find out who was responsible.

In the literature on transaction tracking, the person responsible for misuse of a Work is sometimes referred to as a *traitor,* whereas a person who receives the Work from a traitor is a *pirate*. As this distinction is not meaningful when discussing other applications where piracy is an issue, we do not use this terminology. Instead, we use the term *adversary* to describe anyone who attempts to remove, disable, or forge a watermark for the purpose of circumventing its original purpose.

Transaction tracking is more often called *fingerprinting*, as each copy of a Work can be uniquely identified by the watermark, which is analagous to a human fingerprint that uniquely identifies a person. However, fingerprinting already refers to the detection and recognition of human fingerprints. And fingerprinting is also being used to refer to the passive identification of Works (e.g., broadcast monitoring; see Section 2.1.1). To avoid these ambiguities, we prefer the term *transaction tracking*.

There are few technologies for transaction tracking that do not fall under our definition of watermarking. One common alternative to watermarking is to use visible marks. For example, highly sensitive business documents, such as business plans, are sometimes printed on backgrounds containing large gray digits (see Figure 2.2), with a different number for each copy. Records are then kept about who has which copy. These marks are often referred to as "watermarks"



Watermarking Applications and Their Properties

Ingemar J. Cox, Matt L. Miller and Jeffrey A. Bloom
NEC Research Institute
4 Independence Way
Princeton, NJ 08540
ingemar,mlm,bloom@research.nj.nec.com

Abstract

We describe a number of applications of digital watermarking and then examine the common properties of robustness, tamper-resistance, fidelity, computational cost and false positive rate. We observe that these properties vary greatly depending on the application. Consequently, we conclude that evaluation of a watermarking algorithm is difficult without first indicating the context in which it is to be applied.

1. Introduction

Watermarking methods are often evaluated based on the common properties of robustness, tamper resistance, and fidelity. However, examination of these properties without careful consideration of the application can often be misleading. A watermark designed to serve security needs of the CIA must meet different requirements than one intended for annotating home video. Thus, it is inappropriate to evaluate these two watermarks according to the same standards.

In this paper, we examine how the requirements for watermarking can vary with application. Section 2 briefly describes eight existing and proposed applications of watermarks. This is followed, in Section 3, with a discussion of several properties of watermarking systems, and how their definition and importance depend on the application.

2 Applications

2.1 Broadcast monitoring

In 1997, a scandal broke out in Japan regarding television advertising. At-least two stations had been routinely overbooking airtime. Advertisers were paying for thousands of commercials that were never aired [15]. The practice had remained largely undetected for over twenty years, in part because there were no systems in place to monitor the actual broadcast of advertisements.

There are several types of organizations and individuals interested in broadcast monitoring. Advertisers, of course, want to ensure that they receive the airtime purchased from broadcasting firms.[1] Musicians and actors want to ensure that they receive accurate royalty payments for broadcasts of their performances.[2] And copyright owners want to ensure that their property is not illegally rebroadcast by pirate stations.

We can use watermarks for broadcast monitoring by putting a unique watermark in each video or sound clip prior to broadcast. Automated monitoring stations can then receive broadcasts and look for these watermarks, identifying when and where each clip appears. Commercial systems have been deployed for a number of years and the basic concepts have a long history [14, 3, 18, 10, 4].[3]

2.2 Owner identification

Although a copyright notice is no longer necessary to guarantee copyrights, it is still recommended. The form of the copyright notice is usually Ó *Date, owner*Ó It is visible on books and photographs. Copyright notices can usually be found preceding the content of videotapes and on the packaging of prerecorded video and music products. One disadvantage of such Ohard Ó copyright notices is that they can often be removed from the protected material. Packaging can be lost, video programs can be temporally cropped, and images can be spatially cropped. A digital watermark can be used to provide complementary copyright marking functionality because it becomes an integral part of the content, i.e. the copyright information is embedded in the music to supplement the Ohard Ó notice printed on the packaging.

[1] A recent spot-check by the Screen Actor Guild found an average of $1000 in underpaid royalties per hour of US television programming [2].

[3] The earliest reference we have found [14] is assigned to the Muzak Corporation, famous for providing Oelevator musicÓ and may be the source of the many rumors that Muzak contained subliminal messages.

FIGURE 2.2

Example of a unique identification code printed in the background of a text document.

because they have a physical resemblance to paper watermarks. However, they are not watermarks in our sense of the term, in that we consider imperceptibility to be a defining characteristic. Of course, imperceptible watermarks are preferable to these visible marks for the same reasons watermarks are preferable to textual copyright notices (see Section 2.1.2).

An example of watermarking for transaction tracking was implemented by the now-defunct DiVX Corporation.[4] DiVX sold an enhanced DVD player that implemented a pay-per-view business model. They implemented a variety of security technologies to prevent piracy of their disks, one of which was a watermark designed for transaction tracking. Each DiVX-enabled player would place a unique watermark into every video it played. If someone then recorded that video and started selling copies on the black market, the DiVX corporation could obtain one of the copies and identify the adversary (or, at least, the adversary's DiVX player) by decoding the watermark. To our knowledge, the DiVX watermark was never actually used to trace an adversary before DiVX ceased business.

[4] The demise of DiVX had nothing to do with their watermarking technology.

Another example of transaction tracking is in the distribution of movie dailies. During the course of making a movie, the result of each day's photography is often distributed to a number of people involved in its production. These dailies are highly confidential, yet occasionally a daily is leaked to the press. When this happens, studios quickly try to identify the source of the leak. Studios can use visible text at the edges of the screen to identify each copy of the dailies. However, watermarks are preferable because the text is too easy to remove.

The Academy of Motion Picture Arts and Sciences is responsible for the famous Oscar Awards, which recognize the contributions of actors, actresses, directors, and many others to the art and science of movie making. In the runup to the Oscar ceremonies, the Academy polls all its 5,803 eligible voting members to vote for Best Actor, Best Film, etc. The Academy provides each voting member with Oscar screeners (i.e., copies of all the movies under consideration in either VHS or DVD format), so that members may view all the nominees. However, many of these movies have not yet been released to video. In fact, in some cases, a movie may not have received a broad cinema release. In such circumstances, it may be economically very damaging if a pirated copy of an Oscar screener appears in VHS or DVD format or on the Internet.

In 2004, Technicolor, a division of Thompson, used video watermarking technology licensed from Philips to individually watermark each of the 5,803 voting members' Oscar screeners. After distibuting these copies, pirated video of the films *The Last Samurai, Something's Gotta Give, Big Fish*, and *Mystic River* appeared on the Internet. Analysis of these pirated copies revealed that the original source material had been Oscar screeners provided to the actor Carmine Caridi.[5] Carmine Caridi informed investigators that he had provided the copies to his friend Russell Sprague and was unaware that this friend intended to make illegal copies. Nevertheless, the 70-year-old actor was expelled from the Academy. Columbia Pictures and Warner Bros. Entertainment Ltd. subsequently sued Carmine Caridi for damages. In November 2004, Warner Bros. was awarded the maximum statutory damages of $150,000 per title (i.e., $300,000) in a civil suit.

2.1.5 Content Authentication

It is becoming easier and easier to tamper with digital Works in ways that are difficult to detect. For example, Figure 2.3 shows a modification made to an image using Adobe Photoshop; on the left is the original image; on the right is the modified version. If this image were a critical piece of evidence in a legal case or police investigation, this form of tampering might pose a serious problem. The same problem exists with audio and video.

[5] Carmine Caridi most notably starred in *The Godfather—Part 3*.

FIGURE 2.3

Ease of modifying images: The woman in the image on the left was removed from the scene, resulting in the image on the right. This required about 10 minutes using Adobe Photoshop.

The problem of authenticating messages has been well studied in cryptography [387]. One common cryptographic approach to this problem is the creation of a *digital signature,* which is essentially an encrypted summary of the message. An asymmetric key encryption algorithm is used, so that the key required to encrypt the signature is different from that required to decrypt it. Only the authorized source of messages knows the key required for creating signatures. Therefore, an adversary who tries to change the message cannot create a new signature. If someone subsequently compares the modified message against the original signature, he or she will find that the signatures do not match and will know that the message has been modified.

Digital signature technology has been applied to digital cameras by Friedman [154, 155], who suggests creating a "trustworthy camera" by computing a signature inside the camera. Only the camera would have the key required for creating the signature. Therefore, if we find that a copy of an image matches its signature, we can be sure it is bit for bit identical with the original.

These signatures are metadata that must be transmitted along with the Works they verify. It is thus easy to lose the signatures in normal usage. For example, consider an image authentication system that stores the metadata in a JPEG header field. If the image is converted to another file format that has no space for a signature in its header, the signature will be lost. When a signature is lost, the Work can no longer be authenticated.

A preferable solution might be to embed the signature directly into the Work using watermarking [279]. Epson offers such a system as an option on many of its digital cameras [10, 426, 461]. We refer to such an embedded signature as an *authentication mark*. Authentication marks designed to become invalid after even the slightest modification of a Work are called *fragile* watermarks.

The use of authentication marks eliminates the problem of making sure the signature stays with the Work. Of course, care must be taken to ensure that the act of embedding the watermark does not change the Work enough to make it

appear invalid when compared with the signature. This can be accomplished by separating the Work into two parts: one for which the signature is computed, and one into which the signature is embedded. For example, several authors suggest computing a signature from the high-order bits of an image and embedding the signature in the low-order bits [30, 85].

If a Work containing an authentication mark is modified, the mark is modified along with it. This opens up the possibility of learning more about *how* the Work has been tampered with. For example, if an image is divided into blocks, and each block has its own authentication mark embedded in it, we would be able to gain a rough idea of which parts of the image are authentic and which parts have been modified. This and similar ideas have been suggested by many researchers [30, 447].

An example of where this type of localized authentication might be useful would be in a police investigation of a crime. Imagine that the police receive a surveillance video that has been tampered with. If the video is authenticated with a traditional signature, all the police would know is that the video is inauthentic and cannot be trusted. However, if they use a watermark designed to yield localized authentication, they might discover that each frame of the video is reliable except for a license plate on a car. This would be strong evidence that the identity of someone involved in the crime was removed from the video.

A different type of information that might be learned from examining a modified authentication mark is whether or not lossy compression has been applied to a Work. The quantization applied by most lossy compression algorithms can leave telltale statistical changes in a watermark that might be recognizable. The call for proposals for Phase II of SDMI required this functionality in an audio watermark.

Conversely, we might not care about whether a Work was compressed, and might only be concerned with whether more substantial changes were made, such as that indicated in Figure 2.3. This leads to the field of *semi-fragile* watermarks and signatures, which survive minor transformations, such as lossy compression, but are invalidated by major changes. Techniques for content authentication are discussed in more detail in Chapter 11.

2.1.6 **Copy Control**

Most of the applications of watermarking discussed so far have an effect only after someone has done something wrong. For example, broadcast monitoring helps catch dishonest broadcasters after they fail to air ads that were paid for, and transaction tracking identifies adversaries after they distribute illegal copies. These technologies serve as a deterrent against such wrongdoing. Clearly it is better, however, to actually prevent the illegal action. In the copy control application, we aim to *prevent* people from making illegal copies of copyrighted content.

The first and strongest line of defense against illegal copying is encryption. By encrypting a Work according to a unique key, we can make it unusable to anyone who does not have that key. The key would then be provided to legitimate users in a manner that is difficult for them to copy or redistribute. For example, many satellite television broadcasts are encrypted. The decryption keys are provided to each paying customer on a "smart card," which must be inserted into the customer's television set-top box. Anyone trying to watch or record the broadcast without a smart card would see only scrambled video.

There are three basic ways an adversary might try to overcome an encryption system. The first, and most difficult, is to decrypt the data without obtaining a key. This usually involves some form of search, in which the adversary exhaustively tries decrypting the signal with millions of keys. If the cryptographic system is well designed, the adversary will have to try every possible key. This is impractical if the keys are longer than 50 bits or so.

An easier approach for the adversary is to try to obtain a valid key. This might be done by reverse-engineering hardware or software that contains the key. An example of this approach is embodied in the DeCSS program written by Jon Johansen and two German collaborators [321]. CSS (Content Scrambling System) is the encryption system used to protect DVD video from illegal copying. Johansen and colleagues were able to reverse-engineer a DVD player and find its decryption keys. This allowed them to produce DeCSS, which decrypts any CSS-encrypted video.

By far the easiest method of thwarting encryption is to legally obtain a key and pirate the content after it is decrypted. An adversary who wishes to tape and redistribute satellite broadcasts need only sign up as a customer, acquire a smart card, and connect the output of his or her set-top box to the input of his or her VCR. This points out a central weakness of cryptographic protection: Ultimately, the content must be decrypted before it can be used, and once decrypted, all protection is lost.

In fact, not only must the user decrypt the content, the user must also convert the content to an analog signal in order to perceive it. It is this digital-to-analog conversion that creates what is now referred to as the "analog hole." After conversion to the analog domain, all digital technologies for the protection of content are lost. Subsequent digitization of the analog signal can create a very high-quality illegal digital copy that can then be played and recorded on consumer devices.

What we would like is to somehow allow media to be perceived, yet still prevent it from being recorded. One technology for doing this, in the case of NTSC video, is Macrovision's Videocassette Anticopy Process [355]. This modifies the video signal in such a way as to confuse the automatic gain control on VCRs. The resulting signal is perfectly viewable on a television set, but produces something unwatchable if it is recorded with a VCR. However, this technology applies only to analog television signals. It is not applicable to audio

or to digital signals of any type. Thus, although Macrovision's system is suitable for preventing illegal video recording on existing VCRs, it is not suitable for use in recordable DVDs, digital VCRs, or any other digital video recording technology.

Because watermarks are embedded in the content itself, they are present in every representation of the content and therefore might provide a better method of implementing copy control.[6] If every recording device were fitted with a watermark detector, the devices could be made to prohibit recording whenever a never-copy watermark is detected at its input. This functionality is sometimes referred to specifically as *record control*. Such a system has been envisioned for use on video DVDs by the Copy Protection Technical Working Group (CPTWG), and for use in audio by the SDMI.

There is one substantial nontechnical problem in deploying a copy control system based on watermarking: How do we ensure that every recorder contains a watermark detector? There is no natural incentive for recorder manufacturers to incur the expense of incorporating the detectors in their products. In fact, there is a strong *disincentive,* in that the watermark detector actually *reduces* the value of the recorder from the customer's point of view. The customer would rather have a device that is capable of making illegal copies.

The direct solution to this problem would be to require watermark detectors in recorders by law. However, no such law currently exists, and enacting one would be difficult at best. Furthermore, this legal battle would have to be fought in every country in the world before watermarks could provide worldwide protection. For this reason, neither the CPTWG nor SDMI rely on such laws.

Instead, the CPTWG and SDMI place the requirement for watermark detectors in the patent licenses for technologies that manufacturers *want* to include in their devices. For example, the requirement for a video watermark detector is incorporated into the patent license for CSS. Thus, if manufacturers wish to produce DVD players and recorders that can play CSS-encrypted disks, they will be required by the CSS patent license to include watermark detectors.[7] This approach has the advantage that it relies on existing laws that are well established in most countries.

The patent-license approach has the disadvantage that it allows manufacturers to produce recorders that do not include watermark detectors, as long as they also do not include the desirable technology covered in the patent license.

---

[6] Most of our discussion of copy control with watermarking is taken from Bloom *et al.* [42].

[7] Note that the fact that CSS has been hacked does not diminish its effectiveness as a means of persuading manufacturers to include watermark detectors in their hardware products. Regardless of whether CSS can be illegally decrypted with DeCSS, the manufacturers still require a license to incorporate decryption into their devices. If a manufacturer were to forego the license and just use DeCSS in their product, they would not be able to sell many devices before they were prosecuted for patent infringement.

Thus, there may be perfectly legal DVD player/recorders that implement neither watermark detection nor CSS decryption. Such devices are referred to as *noncompliant,* whereas those that do implement watermarking and decryption are referred to as *compliant*. The existence of noncompliant recorders means that the watermark will not stop all possible illegal recordings.

To counter this, we can use an idea known as *playback control*. When an adversary uses a noncompliant recorder to make an illegal copy of a watermarked Work, that copy will contain the watermark. Compliant players can be required, by patent license, to check for watermarks in the content being played. When a player sees a never-copy watermark, it checks to determine whether it is playing a copy or an original. This can be done in a variety of ways, such as by checking whether the Work is properly encrypted or by looking for a special signature on the media. If it is playing a copy, the player shuts down.

Figure 2.4 illustrates the interaction of encryption, record control, and playback control in a world of legal compliant and noncompliant devices. A legal, encrypted copy of a Work, such as a DVD purchased from a video store, can be played on a compliant player, but not on a noncompliant player, because

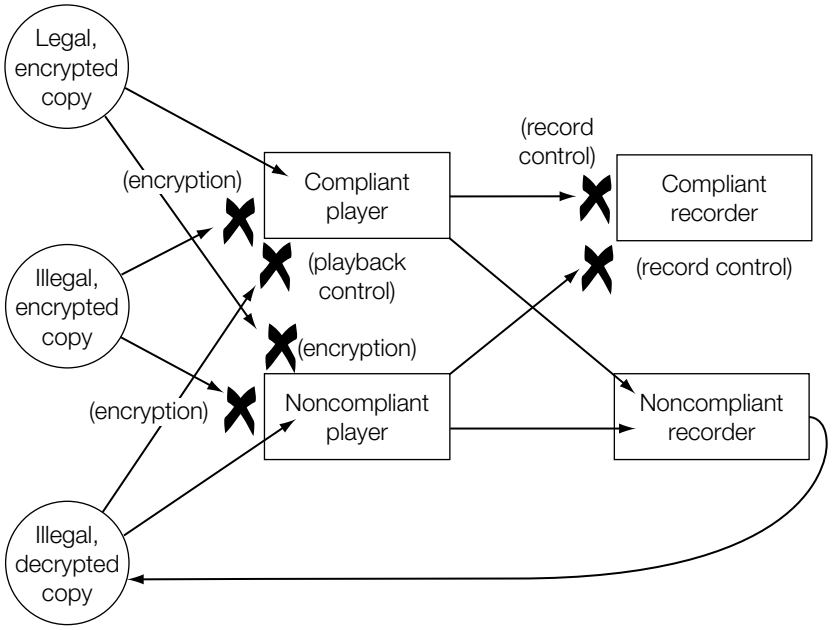


FIGURE 2.4

Relationship between encryption and watermark-based copy control. Arrows indicate potential paths of content. Paths prevented by encryption, record control, or playback control are indicated with X.

the noncompliant player cannot decrypt it. The output of the compliant player cannot be recorded on a compliant recorder, because the recorder would detect the watermark and shut down. However, such output *can* be recorded on a noncompliant recorder, resulting in an unencrypted, illegal copy of the Work. This copy can be played on a noncompliant player, because it is not encrypted, but it cannot be played on a compliant player, because the player would detect the watermark and prohibit playback. Thus, the customer has a choice between buying a compliant device that can play legal, purchased content but cannot play pirated content or a noncompliant device that can play pirated content but not purchased content. The hope is that most customers will choose a compliant option.

The DVD video standard ultimately decided not to introduce watermarking technology into DVD players or recorders. However, the next generation of high-definition (HD) DVD players and recorders incorporate audio and possibly video watermarking technologies. The audio watermark is provided by Verance [2, 3] while the video watermarking, known as VEIL, may be provided by Veil Interactive Technologies.

The VEIL video watermark is a simple method that modulates the intensity of video scan lines [55]. It is discussed further in Section 2.1.7. This video watermark is intended to encode a Rights Assertion Mark (RAM), which is used to support CGMS-A signaling. CGMS-A (Copy Generation Management System Analog) consists of two bits of information representing four states: copy freely, copy no more, copy once, and copy never. The CGMS-A is a video standard that transmits the two bits in the vertical blanking interval of the video signal. As such, it is very easily lost or removed. The RAM is intended to be embedded in video content that contains CGMS-A signaling. If the CGMS-A signal is lost, but the RAM is present, the recording devices are designed not to copy the video signal.

2.1.7 **Device Control**

Copy control falls into a broader category of applications, which we refer to as *device control*. There are several other applications in which devices react to watermarks they detect in content. From the user's point of view, many of these are different from copy control in that they add value to the content, rather than restrict its use.

In 1953, Tomberlin *et al.* [411] described a system to reduce the cost of distributing piped music to offices, stores, and other premises. Traditionally, this music was distributed via land lines (i.e., dedicated telephone lines), which were expensive. MusicScan, the patent assignee, intended to reduce this cost by replacing the dedicated distribution system with a wireless broadcast system. However, it was too costly for this radio transmission to be dedicated to MusicScan's business. Instead, MusicScan premises would receive music from a

commercial radio station that broadcasts commercials and talk as well as music. Tomberlin *et al.*'s invention was to incorporate a watermark signal in the radio transmission that indicated when the radio transmission should be ignored (e.g., when a commercial was being broadcast). This was accomplished with two control signals (embedded as either supersonic or subsonic audio frequencies) that indicated the beginning and end of broadcast segments that were to be blocked.

Another early application of watermarking for control is discussed in a 1981 patent by Ray Dolby [114]. At the time, a number of FM radio stations were broadcasting music using a noise-reduction technique called Dolby FM. To fully exploit Dolby FM, a radio required a corresponding decoder. Listeners had to rely on a listing of stations to determine which were broadcasting Dolby FM signals so that they could manually turn their radio's decoder on or off. Dolby suggested that the radios could be made to automatically turn on their decoders in response to an inaudible tone broadcast within the audio frequency spectrum. Such a tone constitutes a simple watermark.

In 1989, Broughton and Laumeister were awarded a patent [55] for a technique that allows action toys to interact with television programs. In this technique, a simple video watermark modulates the intensities of horizontal scan lines within each field or frame of video. This modulation signal is detected by a light-sensitive device placed near the television. This detector, in turn, transmits a high-frequency infrared signal to interactive devices. In this manner, the actions of a toy can be synchronized with the video being watched on the television. Broughton and Laumeister also mention that modulation of the intensities of the scan lines gives rise to a detectable radio frequency (RF) signal and that this can also form the basis of an alternative detector. An RF detector has the advantage of not being sensitive to ambient lighting conditions and does not need an unrestricted view of the television display.[8]

In a more recent application of watermarking for device control, Digimarc's Mobile system embeds a unique identifier into printed and distributed images such as magazine advertisements, packaging, tickets, and so on. After the image is recaptured by a mobile phone's camera, the watermark is read by the software on the phone and the identifier is used to direct a web browser to an associated web site.

2.1.8 Legacy Enhancement

The situation sometimes arises in which we have a very large, deployed system that we wish to upgrade to provide improved functionality. This upgrade may be incompatible with the existing system. For example, much of the world

[8] This video watermarking system later became known as VEIL and is used in a variety of applications including broadcast monitoring and interactive TV.

is currently transitioning from analog to digital television. This is a costly and time-consuming process. During this transition, completely new digital broadcasting equipment must be introduced and consumers must purchase digital television receivers. Meanwhile, the legacy analog system must continue to function until the vast majority of consumers have transitioned to the digital technology. Eventually, it is intended that analog television broadcasts will cease. In the United States, the analog cutoff date is expected to be 2009, in Japan it is scheduled for 2011, while in the United Kingdom there is a tentative goal of 2012.

Ideally, we would like to upgrade a system such that the new system is backward compatible (i.e., it continues to work with the existing legacy system). Digital watermarking is one of the ways in which this might be accomplished. We briefly describe two proposed systems that illustrate this point.

The international air traffic control system uses an analog communication system to communicate between aircraft and the ground-based air traffic control sites. A protocol exists that requires all pilots to begin any communications by first speaking a call sign (e.g., "This is Lufthansa-two-three-lima-golf" [LH23LG]). There is now interest in upgrading this system to automatically include a machine-readable identifier to authenticate the transmission. Obviously it would be extremely expensive to replace the analog communication systems on all aircraft, from the largest Boeing 777 to the smallest single-seater plane.

Eurocontrol, the European Organisation for the Safety of Air Navigation, has considered whether a digital watermark could be automatically inserted within a pilot's voice communication [186, 187, 194]. By watermarking all voice communications, it is possible to provide a digital identifier such as the aircraft tail number that unambiguously identifies the aircraft. The watermark system is compatibile with the existing communications equipment in both aircraft and air traffic control centers and can therefore be gradually introduced.

Another example is Tektronix's digital watermark encoder for synchronizing audio and video signals [4]. The problem occurs when the video and audio channels of a television signal are processed separately. The digital signal processing introduces different delays to the audio and video channels and may lead to a highly disconcerting phenomenon known as lip-sync, in which it is clear that the motion of the lips is either ahead or behind the speech. The Tektronix product embeds a highly compressed version of the audio signal within the video signal, prior to any digital signal processing. After all signal processing is completed, the real-time audio signal is compared to the embedded audio signal in order to determine if any time delay has been introduced. If so, this delay can be removed prior to broadcasting.

A similar problem may occur when MP3 players attempt to display the lyrics of songs in synchrony with the music. One solution, pioneered by MarkAny of Korea, is to embed the lyrics directly into the audio signal using watermark technology [5]. The technology, known as MediaSync, is very popular in Korea.

2.2 APPLICATIONS OF STEGANOGRAPHY

There are many reasons why two parties might wish to secretly communicate. The reason may be benign, as for example, in the case of two lovers who wish to conceal their relationship. Or it may be political, as in the case of a dissident organization wishing to communicate among themselves or with a forbidden organization outside their country. Or it may be criminal, as in the case of organized crime or terrorism.

There are probably many other applications of steganography, and the authors welcome readers to provide us with more examples. Of course, successful steganography is not detectable, so many of the most successful applications may never become public.

An interesting case of steganography was described in the early 1980s by Simmons [375]. According to the disarming treaty SALT, the United States and the Soviet Union mutually agreed to place sensors in their nuclear facilities that would communicate the number of missiles but not other information, such as the location. To prevent unauthorized tampering with the transmitted signals, the communication was protected using standard digital signatures. Both countries quickly became concerned about the other party smuggling additional information through so-called subliminal channels that existed in most digital signature schemes at that time. This prompted research in developing subliminal-free digital signatures [112].

Here we examine two applications of steganography, namely covert communications by dissidents and covert communications by criminals. These two applications are highlighted in order to examine the very different environmental conditions in which they operate.

2.2.1 Steganography for Dissidents

There are many countries in the world where political dissent is neither tolerated nor legal. Any dissident organization must therefore exercise extreme caution when communicating among each other or with international organizations such as Amnesty International. The environment that dissidents operate in is usually overtly hostile. As such, dissidents are aware that they are under surveillance, and the adversary (i.e., the reigning government), is extremely powerful and possibly ruthless. Any restraint shown by the adversary is only due to pressure from international organizations.

Broadly, there are three ways that dissidents might conceal their communications: encryption, anonymous remailers, and steganography.

Encryption can provide dissidents with a means to assure the privacy of their communications. And if a sufficiently large key is used, then the likelihood of decryption by technical means is negligible, irrespective of the resources available to the adversary. Unfortunately, the very fact that two people are

exchanging encrypted messages indicates that they have something to conceal. An adversary may therefore decide to arrest the two dissidents and obtain the decryption key through torture.

An alternative solution might be to use an anonymous remailer [74]. As the name suggests, anonymous remailers permit two people to communicate anonymously (i.e., Alice and Bob may not even be aware of each other's identity). However, more important, an adversary monitoring transmissions (e.g., Internet packets), from Alice, is unable to trace them once they enter the remailer. Thus, the adversary is unaware of who Alice is communicating with. Unfortunately, like encryption, the use of an anonymous remailer is a clear sign that Alice has something to hide. Once again, this may be sufficient for the adversary to arrest and interrogate Alice. However, another option is also open to the adversary. Since the IP addresses of the anonymous remailers are known, the adversary may decide to block all communications from their country to these addresses. This is possible since the government has total control of the national communications network.

Steganography may therefore be the safest form of communications between dissidents. Benign communications within the national population are unlikely to be forbidden. Furthermore, basic communications are essential for the day-to-day operation of any modern economy. Thus, individuals who can legitimately engage in permitted communications might also use this communications channel to engage in covert communications as well.

Note that in this covert communications environment, the communicating parties, Alice and Bob, are aware that they are being monitored and the adversarial State often wishes to publicize its surveillance in order to deter potential dissidents.

2.2.2 **Steganography for Criminals**

In a democracy, a State's ability to monitor its citizens is significantly restrained by national laws. For example, in the United States, wiretapping (i.e., the monitoring of an individual's phone calls), is only permitted with a federal judge's order. Within this operating environment, an individual is usually unaware that he or she is under surveillance, although he or she may assume as much, and act accordingly. Furthermore, the adversary (e.g., the FBI), usually wishes to conceal any surveillance operation. In this environment, let us reconsider the options for communications.

Encryption can once again be used to guarantee the privacy of communications between Alice and Bob. However, a primary objective of surveillance is to identify all members of the criminal organization. Thus, unless Alice uses encryption for *all* communications, a highly unlikely situation, then the FBI can reliably infer the identitiy of co-conspirators by simply identifying the recipients of encrypted communications.

Anonymous remailers, however, offer a viable alternative for a criminal organization to communicate among its members.[9] First, the use of anonymous remailers is not illegal and the FBI does not have the authority to deny access to such remailers. The advantage of this approach is that no knowledge of the recipients' identities is obtained. Thus, in theory, anonymous remailers offer maximal protection.

Anonymous remailers rely on information being securely stored on the remailers, which indirectly hold the information needed to direct the mail to its intended recipient. In practice, there have been a number of occasions when law enforcement agencies have demanded access to this information [8], even when the remailers are located in other countries. Thus, in practice, there is some risk that anonymous remailers cannot guarantee anonymity.[10] We note that more recent designs [106] alleviate the risk of legal subpoenas.

Steganography offers an alternative to anonymous remailers. A possible disadvantage of steganography is that the communicating parties must have some stego program installed on their computers, which may put them in a compromising situation if the computer is seized. Even if the stego program is uninstalled, many applications still leave footprints behind that can be traced and used to prove that a specific stego application was previously installed.[11] Some steganographic methods are, however, so easy that the embedding process can be run directly as a command line and there is no need to have any stego software installed.

2.3 PROPERTIES OF WATERMARKING SYSTEMS

Watermarking systems can be characterized by a number of defining properties [97, 180, 329, 448]; in this section we highlight 10 of them. The relative importance of each property is dependent on the requirements of the application and the role the watermark will play. In fact, even the interpretation of a watermark property can vary with the application.

We begin by discussing a number of properties typically associated with a watermark embedding process: *effectiveness, fidelity,* and *payload*. We then

[9] Note that anonymous remailers use encryption.

[10] Anonymous remailers can also be used to anonymize voice communications when phone calls are conducted using voice-over IP (VOIP). However, it was demonstrated that the IP packets originating from Alice could be watermarked (by adding a known pseudo random sequence of delays to the packets) [435]. Then a number of suspected destinations could be monitored and the watermark recognized, thereby identifying the receiver of the phone call.

[11] Wetstone's Gargoyle, *http://www.wetstonetech.com*, is an example of a commercially available application that can detect uninstalled programs.

turn to properties typically associated with detection: *blind* or *informed detection, false positive behavior,* and *robustness*. The next two properties, *security* and the use of secret *keys,* are closely related in that the use of keys is usually an integral part of any security feature inherent in a watermarking scheme. We then discuss the ability to change the message encoded by a watermark either by modifying the watermark itself or by representing the message with multiple watermarks. The section concludes with a discussion of the various *costs* associated with both watermark embedding and watermark detection.

2.3.1 **Embedding Effectiveness**

We define a watermarked Work as a Work that when input to a detector results in a positive detection. With this definition of watermarked Works, the *effectiveness* of a watermarking system is the probability that the output of the embedder will be watermarked. In other words, the effectiveness is the probability of detection immediately after embedding. This definition implies that a watermarking system might have an effectiveness of less than 100%.

Although 100% effectiveness is always desirable, this goal often comes at a very high cost with respect to other properties. Depending on the application, we might be willing to sacrifice some effectiveness for better performance with respect to other characteristics. As an example, consider a stock photo house that needs to embed *proof of ownership* watermarks in thousands of images each day. Such a system might have a very high fidelity requirement, and it may be the case that certain images cannot be successfully watermarked within those fidelity constraints. The photo house may then have to decide whether to allow the images to remain unmarked, and thus unprotected, or allow the introduction of more distortion to maintain a 100% effectiveness rate. In some cases, the former choice is preferable.

In some cases, the effectiveness of a watermarking system may be determined analytically. An example of this is presented in Section 7.6.2. It can also be estimated empirically by simply embedding a watermark in a large test set of images. The percentage of output images that result in positive detections will approximate the probability of effectiveness, provided the number of images in the set is sufficiently large and is drawn from the same distribution as the expected application images.

2.3.2 **Fidelity**

In general, the *fidelity* of a watermarking system refers to the perceptual similarity between the original and watermarked versions of the cover Work. However, when the watermarked Work will be degraded in the transmission process prior to its being perceived by a person, a different definition of fidelity may be more appropriate. We may define the fidelity of a watermarking system as the perceptual similarity between the unwatermarked

and watermarked Works at the point at which they are presented to a consumer.

Consider the case of watermarked video that will be transmitted using the NTSC broadcast standard, or audio that will be transmitted over AM radio. The quality of these broadcast technologies is relatively low, so the differences between the original and watermarked Works may become imperceptible after the channel degradations. Conversely, in HDTV and DVD video and audio, the signals are very high in quality, and require much higher fidelity watermarks.

In some applications, we can accept mildly perceptible watermarks in exchange for higher robustness or lower cost. For example, Hollywood dailies are not finished products. They are usually the result of poor transfers from film to video. Their only purpose is to show those involved in a film production the raw material shot so far. A small visible distortion caused by a watermark will not diminish their value.

2.3.3 Data Payload

Data payload refers to the number of bits a watermark encodes within a unit of time or within a Work. For a photograph, the data payload would refer to the number of bits encoded within the image. For audio, data payload refers to the number of embedded bits per second that are transmitted. For video, the data payload may refer to either the number of bits per field (or frame) or the number of bits per second. A watermark that encodes N bits is referred to as an *N-bit watermark*. Such a system can be used to embed any one of 2^N different messages.

Different applications may require very different data payloads. Copy control applications may require just 4–8 bits of information to be received over a period of, say, every 10 seconds for music and perhaps 5 minutes for video. In such scenarios, the data rate is approximately 0.5 bits per second for music and 0.02 bits per second for video. In contrast, television broadcast monitoring might require at least 24 bits of information to identify all commercials, and this identification may need to occur within the first second of the start of the broadcast segment. Thus, the data rate in this application is 24 bits per second, over three orders of magnitude larger than for a video copy control application.

Many applications require a detector to perform two functions. It must determine if a watermark is present and, if so, identify which of the 2^N messages is encoded. Such a detector would therefore have $2^N + 1$ possible output values: 2^N messages plus "no watermark present."

In the watermarking research literature, many systems have been proposed in which there is only one possible watermark and the detector determines whether or not that watermark is present. These have sometimes been referred to as one-bit watermarks because there are 2^1 possible outputs: watermark present and watermark absent. However, this is inconsistent with the

previously described naming convention. We prefer to call these zero-bit watermarks because they have $2^0 + 1 = 2$ possible output values.

2.3.4 Blind or Informed Detection

In some applications, the original, unwatermarked Work is available during detection. For example, in a transaction-tracking application, it is usually the owner of the original Work who runs the detector, in order to discover who illegally distributed a given copy. The owner, of course, should still have an unwatermarked version of the Work and can thus provide it to the detector along with the illegal copy. This often substantially improves detector performance, in that the original can be subtracted from the watermarked copy to obtain the watermark pattern alone. The original can also be used for registration, to counteract any temporal or geometric distortions that might have been applied to the watermarked copy (see Chapter 9).

In other applications, detection must be performed without access to the original Work. Consider a copy control application. Here, the detector must be distributed in every consumer recording device. Having to distribute the unwatermarked content to every detector would not only be impractical, it would defeat the very purpose of the watermarking system.

We refer to a detector that requires access to the original, unwatermarked Work as an *informed detector*. This term can also refer to detectors that require only some information derived from the original Work, rather than the entire Work. Conversely, detectors that do not require any information related to the original are referred to as *blind detectors*. Whether a watermarking system employs blind or informed detection can be critical in determining whether it can be used for a given application. Informed detection can only be used in those applications where the original Work is available.

In the watermarking literature, systems that use informed detection are often called *private watermarking systems,* whereas those that use blind detection are called *public watermarking systems*. This terminology refers to the general usefulness of the systems in applications in which only a select group of people may detect the watermark (private watermarking applications) or applications in which everyone must be allowed to detect the watermark (public watermarking applications). In general, the original Works are only available in private applications, and therefore informed detectors cannot be used in public applications. However, for reasons discussed in Chapter 10, we usually avoid the use of "public" and "private."

2.3.5 False Positive Rate

A *false positive* is the detection of a watermark in a Work that does not actually contain one. When we talk of the false positive rate, we refer to the number of false positives we expect to occur in a given number of runs of the detector.

Equivalently, we can discuss the probability that a false positive will occur in any given detector run. There are two subtly different ways to define this probability, which are often confused in the literature. They differ in whether the watermark or the Work is considered the random variable.

In the first definition, the false positive probability is the probability that given a fixed Work and randomly selected watermarks, the detector will report that a watermark is in that Work. The watermarks are drawn from a distribution defined by the design of a watermark generation system. Typically, watermarks are generated by either a bit-encoding algorithm or by a Gaussian, independent random number generator. In many cases, the false positive probability, according to this first definition, is actually independent of the Work, and depends *only* on the method of watermark generation.

In the second definition, the false positive probability is the probability that given a fixed watermark and randomly selected Works, the detector will detect that watermark in a Work. The distribution from which the Work is chosen is highly application dependent. Natural images, medical images, graphics, music videos, and surveillance video all have very different statistics. The same is true of rock music, classical music, and talk radio. Moreover, while these distributions are different from one another, they are also likely to be very different from the statistics of the watermark generation system. Thus, false positive probabilities based on this second definition can be quite different from those based on the first definition. Experimental results, presented in Chapter 7, bear this out.

In most applications, we are more interested in the second definition of false positive probability than in the first. However, in a few cases, the first definition is also important, such as in the case of transaction tracking, in which the detection of a random watermark in a given Work might lead to a false accusation of theft.

The required false positive probability depends on the application. In the case of proof of ownership, the detector is used so rarely that a probability of 10^{-6} should suffice to make false positives unheard of. On the other hand, in a copy control application, millions of watermark detectors are constantly being run on millions of Works all over the world. If one unwatermarked Work consistently generates false positives, it could cause serious trouble. For this reason, the false positive rate should be infinitesimal. For example, the general consensus is that watermark detectors for DVD video should have a false positive probability of 1 in 10^{12} frames [42], or about 1 in 1,000 years of continuous operation.

2.3.6 Robustness

Robustness refers to the ability to detect the watermark after common signal processing operations. Examples of common operations on images include

spatial filtering, lossy compression, printing and scanning, and geometric distortions (rotation, translation, scaling, and so on). Video watermarks may need to be robust to many of the same transformations, as well as to recording on video tape and changes in frame rate, among other influences. Audio watermarks may need to be robust to such processes as temporal filtering, recording on audio tape, and variations in playback speed that result in wow and flutter.

Not all watermarking applications require robustness to all possible signal processing operations. Rather, a watermark need only survive the common signal processing operations likely to occur between the time of embedding and the time of detection. Clearly this is application dependent. For example, in television and radio broadcast monitoring, the watermark need only survive the transmission process. In the case of broadcast video, this often includes lossy compression, digital-to-analog conversion, analog transmission resulting in low-pass filtering, additive noise, and some small amount of horizontal and vertical translation. In general, watermarks for this application need not survive rotation, scaling, high-pass filtering, or any of a wide variety of degradations that occur only prior to the embedding of the watermark or after its detection. Thus, for example, a watermark for broadcast monitoring need not be robust to VHS recording.

In some cases, robustness may be completely irrelevant, or even undesirable. In fact, an important branch of watermarking research focuses on fragile watermarks. A fragile watermark is one designed so that it is *not* robust. For example, a watermark designed for authentication purposes should be fragile. Any signal processing application applied to the image should cause the watermark to be lost. Fragile watermarking is discussed in Chapter 11.

At the other extreme, there are applications in which the watermark must be robust to every conceivable distortion that does not destroy the value of the cover Work. This is the case when the signal processing between embedding and detection is unpredictable, or when the watermark must be secure against hostile attack (see next section). Robust watermarking is discussed in Chapter 9.

2.3.7 **Security**

The *security* of a watermark refers to its ability to resist hostile attacks. A hostile attack is any process specifically intended to thwart the watermark's purpose. The types of attacks we might be concerned about fall into three broad categories:

- Unauthorized removal
- Unauthorized embedding
- Unauthorized detection

Unauthorized removal and embedding are referred to as *active* attacks because these attacks modify the cover Work. Unauthorized detection does not modify the cover Work and is therefore referred to as a *passive* attack.

The relative importance of these attacks depends on the application. In fact, there are situations in which the watermark has no hostile enemies and need not be secure against *any* type of attack. This is usually the case when a watermark is used to provide enhanced functionality to consumers. However, for applications that do require some level of security, it is important to understand the distinctions between these types of attack.

Unauthorized removal refers to attacks that prevent a Work's watermark from being detected. It is common to distinguish between two forms of unauthorized removal: *elimination* attacks and *masking* attacks. The distinction is subtle and is discussed in detail in Chapter 10. Intuitively, *elimination* of a watermark means that an attacked Work cannot be considered to contain a watermark *at all*. That is, if a watermark is eliminated, it is not possible to detect it even with a more sophisticated detector. Note that eliminating a watermark does *not* necessarily mean reconstructing the original, unwatermarked Work. Rather, the goal of the adversary is to make a new Work that is perceptually similar to the original, but will never be detected as containing a watermark. The original, itself, would satisfy this goal, but it is only one of many Works that would.

Masking of a watermark means that the attacked Work can still be considered to contain the watermark, but the mark is undetectable by existing detectors. More sophisticated detectors might be able to detect it. For example, many image watermark detectors cannot detect watermarks that have been rotated slightly. Thus, an adversary may apply a rotation that is slight enough to be unnoticeable, with the distorted image therefore having acceptable fidelity. Because the watermark detector is sensitive to rotations, the watermark will not be detected. Nevertheless, the watermark could still be detected by a more sophisticated detector capable of correcting for the rotation. Therefore, we can think of the watermark as still being present.

One interesting form of unauthorized removal is known as a *collusion* attack. Here, the attacker obtains several copies of a given Work, each with a different watermark, and combines them to produce a copy with no watermark. This is primarily a concern in transaction tracking, which entails putting a different watermark in each copy. With existing watermarking systems, it is generally believed that a fairly small number of copies suffices to make a collusion attack successful [130, 236, 389]. How serious this problem is depends on the context in which transaction tracking is being used. In the DiVX application described in Section 2.1.4, we can easily imagine that an adversary might obtain a dozen players and thus be able to remove the watermark. However, in the studio dailies application, it is very unlikely that any one employee will be able to obtain many different copies of a given film clip; therefore, collusion attacks are of less concern.

Unauthorized *embedding,* also referred to as *forgery,* refers to acts that embed illegitimate watermarks into Works that should not contain them. For example, consider an authentication application, discussed in Section 2.1.5. Here, we are not concerned with whether an adversary can render a watermark undetectable, in that doing so would cause the detector to, correctly, identify a Work as inauthentic. However, if an adversary has the ability to perform unauthorized embedding, she can cause the detector to falsely authenticate an invalid Work.

Unauthorized *detection,* or *passive* attacks, can be broken down into three levels of severity. The most severe level of unauthorized detection occurs when an adversary detects and deciphers an embedded message. This is the most straightforward and comprehensive form of unauthorized reading. A less severe form of attack occurs when an adversary can detect watermarks and distinguish one mark from another, but cannot decipher what the marks mean. Because watermarks refer to the Works in which they are embedded, the adversary might be able to divine the meanings of the marks by comparing them to their cover Works. The least severe form of attack occurs when an adversary is able to determine that a watermark is present, but is neither able to decipher a message nor distinguish between embedded messages.

In general, passive attacks are of more concern in steganography than in watermarking, but there are watermarking applications in which they might be important. For example, suppose we have a broadcast monitoring company that embeds watermarks for free, and then charges for the monitoring reports. An adversary who can read our watermarks could set up a competing service, without having to incur the cost of embedding.

Chapter 10 provides a framework for analyzing the security requirements of watermarking applications and discusses the difficulty of creating secure watermarks under various conditions. It also describes a few known, general attacks, and ways to counter (some of) them.

2.3.8 Cipher and Watermark Keys

In cryptography, security is usually provided by the use of secret *keys*. These can be thought of as more or less arbitrary sequences of bits that determine how messages are encrypted. In the simplest systems, a message encrypted with a given key can only be decrypted with the same key. Many watermarking systems are designed to use keys in an analogous manner. In such systems, the method by which messages are embedded in watermarks depends on a key, and a matching key must be used to detect those marks.

Before the introduction of keys, the security of most cryptographic algorithms relied on keeping those algorithms secret. This led to two problems. First, if the security of an algorithm was compromised, a completely new algorithm had to be developed. Second, because of the need for secrecy, it was not possible to disclose an algorithm to outside researchers for study. This meant

that many security weaknesses were never discovered before the algorithms were actually in use. These problems are now largely solved by using keys.

In modern cryptographic algorithms, security is derived from securing only the key, not the entire algorithm. Thus, encryption and decryption algorithms can be publicly disclosed and evaluated using one key. A different key can then be employed during actual use. Finally, if this key is compromised, it is not necessary to change the entire algorithm; only a new secret key must be selected.[12] Thus, a well-designed cipher should meet the following standards:

- Knowledge of the encryption and decryption algorithms should not compromise the security of the system.
- Security should be based on the use of keys.
- Keys should be chosen from a large *keyspace* so that searching over the space of all possible keys is impractical.

It is desirable to apply the same standards to watermarking algorithms. Unfortunately the keyspace anology between cryptography and watermarking is flawed. In particular, in cryptography, all n-bits of a cipher key must be determined, and no useful information is provided by similar keys (i.e., keys that only differ in a few bits). However, in watermarking, an adversary often needs only to find a key that is close, significantly reducing the effective size of the keyspace [98]. Further, the security requirements for watermarks are often different from those for ciphers. Therefore, we cannot simply adapt cryptographic methods to watermarking. Cryptography is only concerned with the prevention of unauthorized reading and writing of messages. It can therefore be used in watermarking to prevent certain forms of passive attack and forgery, but it does not address the issue of watermark removal. This problem is analogous to the problem of signal jamming in military communications, and so it is to that field we must turn to find a possible solution.

To provide resistance to jamming and interference, the military developed *spread spectrum* communications, in which a narrowband signal is spread over a much larger bandwidth. The exact form of the spreading is a secret known only by the transmitter and receivers. Without knowledge of the spreading function, it is almost impossible for an adversary to detect or interfere with a transmission. This spreading function is also based on a secret analogous to a key in cryptography (although it is seldom, if ever, referred to as such).

[12] An important issue in any key-based system is that of *key management,* the procedures used to maintain secrecy of keys and replace those that have been compromised. This topic, however, is outside the scope of this book. The reader is directed to [23, 362, 387] for further information.

Watermark algorithms can be designed to use secret keys in a manner similar to that in spread spectrum. For example, one simple form of watermarking algorithm, described in Chapter 3, adds a pseudo-random noise pattern (or *PN pattern*) to an image. For the system to work, the embedder and detector must use the same PN pattern. Thus, the PN pattern, or a seed used to generate the PN pattern, can be considered a secret key.

Ideally, it should not be possible to detect the presence of a watermark in a Work without knowledge of the key, even if the watermarking algorithm is known. Further, by restricting knowledge of the key to only a trusted group (i.e., by preventing an adversary from learning the key), it should become extremely difficult, if not impossible, for an adversary to remove a watermark without causing significant degradation in the fidelity of the cover Work.

Because the keys used during embedding and detection provide different types of security from those used in cryptography, it is often desirable to use *both* forms of keys in a watermarking system. That is, messages are first encrypted using one key, and then embedded using a different keys. To keep the two types of keys distinct, we use the terms *cipher key* (or crypto key) and *watermark key,* respectively. The relationship among cryptography, spread spectrum communications, and watermarking is discussed in more detail in Chapter 10.

2.3.9 Modification and Multiple Watermarks

When a message is embedded in a cover Work, the sender may be concerned about message modification. Whereas the ability to modify watermarks is highly undesirable in some applications, there are others in which it is necessary. For example, American copyright law grants television viewers the right to make a single copy of broadcasted programs for time-shifting purposes (i.e., you are permitted to make a copy of a broadcast for the noncommercial purpose of watching that broadcast at a later time). However, you are not permitted to make a copy of this copy. Thus, in copy control applications, the broadcasted content may be labeled copy-once and, after recording, should be labeled copy-no-more. This modification of the recorded video can be accomplished in a variety of ways. The most obvious manner is to simply alter the embedded watermark denoting copy-once so that it now denotes copy-no-more. However, if a watermark is designed to permit easy modification, there is the risk that illegal modification will become commonplace. Imagine a device or program that modifies the video from copy-once to copy-freely.

An alternative to altering the existing watermark is to embed a second watermark. The presence of both watermarks can then be used to denote the copy-no-more state. Conversely, the initial copy-once state can be denoted by two watermarks, the primary and secondary. The secondary watermark could be a relatively fragile watermark compared to the primary one, and its removal would denote copy-no-more. These two methods appear at first sight

to be equivalent but in fact have very different false positive behaviors. Their implementation also raises technical challenges, such as the need to embed watermarks in compressed media without changing the bit rate [178, 340].

Another example of the need for multiple watermarks to coexist in a Work arises in the area of transactional watermarks. Content distribution often includes a number of intermediaries before reaching the end user. Thus, a record label might first want to include a watermark identifying itself as the copyright owner. The Work might then be sent to a number of music web sites. Each copy of the Work might have a unique watermark embedded in it to identify each distributor. Finally, each web site might embed a unique watermark in each Work it sells for the purpose of uniquely identifying each purchaser.

2.3.10 Cost

The economics of deploying watermark embedders and detectors can be extremely complicated and depends on the business models involved [107]. From a technological point of view, the two principal issues of concern are the speed with which embedding and detection must be performed and the number of embedders and detectors that must be deployed. Other issues include whether the detectors and embedders are to be implemented as special-purpose hardware devices or as software applications or plugins.

In broadcast monitoring, both embedders and detectors must work in (at least) real time. This is because the embedders must not slow down the production schedule, and the detectors must keep up with real-time broadcasts. On the other hand, a detector for proof of ownership will be valuable even if it takes days to find a watermark. Such a detector will only be used during ownership disputes, which are rare, and its conclusion about whether the watermark is present is important enough that the user will be willing to wait.

Furthermore, different applications require different numbers of embedders and detectors. Broadcast monitoring typically requires a few embedders and perhaps several hundred detectors at different geographic locations. Copy control applications may need only a handful of embedders but millions of detectors embedded in consumer video and audio devices. Conversely, in the transaction-tracking application implemented by DiVX, in which each player embeds a distinct watermark, there would be millions of embedders and only a handful of detectors. In general, the more numerous a device needs to be for a given application the less it must cost.

2.4 EVALUATING WATERMARKING SYSTEMS

Most people who deal with watermarking systems need some way of evaluating and comparing them. Those interested in applying watermarking to

an application need to identify the systems that are most appropriate. Those interested in developing new watermarking systems need measures by which to verify algorithmic improvements. Such measures may also lead to ways of optimizing various properties.

2.4.1 The Notion of "Best"

Before we can evaluate a watermarking system, we need to have some idea of what makes one system better than another, or what level of performance would be best. If we are interested in using a watermark for some specific application, our evaluation criteria must depend on that application. For example, if we are evaluating a video watermark for use in copy control, we might want to test its robustness to small rotations, in that these could be used as an attack (see Section 2.3.7). However, such robustness might be irrelevant for broadcast monitoring, because rotations are unlikely to occur during normal broadcasting, and we might not be concerned with security against active attacks.

If we are interested in testing the merit of a new watermarking system in comparison to existing systems, we have more flexibility in choosing our test criteria. In general, by showing that a new system provides an improvement in any one property, all else being equal, we can show that the system is worthwhile, at least for applications in which that property is important.

Moreover, an improvement in one property can often be traded for improvements in others. For example, suppose we have a watermarking system that carries a data payload of 20 bits, and we devise a way of extending this to 30 bits, without changing the false positive probability, robustness, or fidelity. Clearly, the new system would be better than the old for applications that can use greater data payload. However, for other applications, we can easily trade this higher payload for a lower false positive probability. This could be done by modifying the embedder to take only 20-bit messages, but to embed a 10-bit checksum along with each message. The detector, then, would extract all 30 bits and check whether the checksum is correct, declaring that no watermark is embedded if the check fails. This would result in a 20-bit system with a false positive probability 1,024 times lower than the original.

Furthermore, many watermark detectors employ a *detection threshold* parameter, which we can lower to translate our improved false positive probability into improved robustness. Further still, many embedders employ an *embedding strength* parameter, which trades robustness for fidelity. Thus, if we can verify that a new system carries larger payloads than an older one, we can often claim that it will be better for a wide variety of applications.

2.4.2 Benchmarking

Once the appropriate test criteria have been determined, we can turn our attention to developing a test. If we are evaluating watermarks for a specific

application, each criterion can be laid out as a minimum requirement for a relevant property, and a suite of tests can be developed to measure whether systems meet those requirements. All tests of a given system must be performed using the same parameter settings at the embedder and detector (e.g., a constant embedding strength and detection threshold). An early example of such a testing program was the CPTWG's effort to test watermarks for copy control in DVD recorders [34] (see Section 2.1.6).

For purposes of watermarking research, there has been some interest in the development of a universal benchmark. This benchmark could be used by a researcher to assign a single, scalar "score" to a proposed watermarking system. The score could then be used to compare it against other systems similarly tested.

A proposed benchmark for image watermarking systems [247] specifies a number of bits of data payload that must be embedded (80), along with a level of fidelity that must be maintained, as measured by a specific perceptual model (see Chapter 8 for more on perceptual modeling). Holding these two properties constant, then, the robustness of the system to several types of distortion is tested using a program called *Stirmark* [328]. This program applies distortions that have little effect on the perceptual quality of images, but are known to render most watermarks undetectable. The system's performance in these tests is combined into a single score.

Unfortunately, this benchmark can only be performed on certain watermarking systems and is only relevant to certain watermarking applications. Many watermarking systems are designed to carry very small data payloads (on the order of 8 bits or fewer), with very high robustness, very low false positive probability, and/or very low cost. Such systems typically employ codes that cannot be expanded to 80 bits (see Chapter 4). Furthermore, the tests performed by Stirmark are not critical to many applications that do not have stringent security requirements. They also do not represent a comprehensive test of the security required in applications in which an adversary is likely to have a watermark detector (see Chapter 10).

We believe that no benchmark can ever be relevant to all watermarking systems and applications. Thus, in this book, we test our example algorithms with experiments designed specifically for the points the algorithms illustrate. Almost every algorithm is a modification of one previously presented and is based on a theoretical discussion that suggests the modification should yield improved performance in some particular property. Typically, we try to hold as many other properties constant as possible while we test the effects of the modification.

2.4.3 Scope of Testing

In general, watermarking systems should be tested on a large number of Works drawn from a distribution similar to that expected in the application. For example, we would not necessarily expect an algorithm that was tuned to

"Lena"[13] to be ideally suited for X-ray images, satellite photos, or animation frames. If a system is being tested without a specific application in mind, the Works it is tested on should be representative of a typical range of applications.

These two issues—size and typicality of the test set—are especially important in testing false positive rates. Imagine a watermark detector that will be examining a large number of Works, looking for one particular watermark pattern. If this system is required to have a false positive rate of 10^{-6}, we would expect to see, on average, no more than one false positive in every one million Works examined. To truly verify this performance, we would need to present the detector with many millions of Works. Of course, in many cases, such a test might be infeasible, and we have to make do with tests that verify some statistical model used to predict the false positive rate (see Sections 7.2 and 7.6.1). Even these tests, though they do not require millions of Works, can be misleading if performed on too few.

For all of the experiments in this book, we started with a database of 20,000 stock images [86], at a resolution of 240 × 368 pixels. The majority of these images are photographs, but some of them are paintings and textures. Most of the experiments were run on 2,000 images chosen randomly from this set. Some false positive tests were performed on the full database.

2.5 PROPERTIES OF STEGANOGRAPHIC AND STEGANALYSIS SYSTEMS

The primary goal of steganography is to hide the fact that a covert communication is present within an innocuous communication. And the primary goal of steganalysis is to detect when a covert communication is occurring. In contrast, in many digital watermarking applications, knowledge that the watermark is present, though imperceptible, is common. In fact, such knowledge is considered desirable, since it may act as a deterrent to illegal use.

While steganographic and digital watermarking algorithms can be built on a shared foundation of data-hiding principles, the desired properties of steganographic and steganalysis systems are quite different from those of digital watermarking. Below, we reexamine these properties and introduce additional properties that are often more critical.

2.5.1 Embedding Effectiveness

The embedding effectiveness of a watermarking system measures the probability of an error when the detector is applied immediately after embedding

[13] "Lena" is the common name for the image of Lena Sjööblom, discussed in Section 2.1.2. It is perhaps the most-often-used test image in image processing.

but prior to any subsequent distortion. The fact that this error rate may be non-zero reflects the fact that constraints imposed on the embedder (e.g., fidelity requirements), may prevent successful embedding of all the bits.

This property is absent from discussion of steganography, despite the fact that constraints are also imposed on the steganographic embedder, e.g., undetectability (see Section 12.3). One reason this is not considered is that the embedder is free to choose the most appropriate cover Work. Thus, if constraints do not permit the embedding of a covert message in one particular cover Work, the steganographic embedder is free to use an alternative cover Work. This flexibility is not available to a watermark embedder, which has no control over the choice of cover Work.

2.5.2 Fidelity

At first sight, it is somewhat counterintuitive that fidelity is also not a primary property of a steganographic system. After all, if we can see a change, surely we can detect the presence of a covert message. Unfortunately, steganalysis systems do not have access to the original cover Work. Thus, if an original image of a man in a blue suit is altered to one of a man in a gray suit, then, provided the stego Work meets the constraints on undetectability (see Section 12.3), the adversary has no means to know that a covert message is present.

This example highlights the fact that for steganography, the cover Work is of no value. It simply serves to hide the covert message. This is very different from digital watermarking, for which the cover Work is of primary importance and the fidelity of the cover Work is preeminent.

2.5.3 Steganographic Capacity, Embedding Capacity, Embedding Efficiency, and Data Payload

The steganographic capacity is equivalent to a watermark's data payload. The *embedding capacity* is the maximum number of bits that can be hidden in a given cover Work. For example, if we embed a covert message by modifying the least significant bit (LSB) of grayscale images, the embedding capacity (in bits) is the number of pixels in the image. *Steganographic capacity* is the maximum number of bits that can be hidden in a given cover Work, such that the probability of detection by an adversary is negligible. The steganographic capacity is therefore likely to be much less than the embedding capacity. Determining the steganographic capacity is a very difficult task even for the simplest embedding schemes.

Most steganographic schemes can avoid being detected by current steganalysis simply by decreasing the amount of information embedded in a cover Work. Alternatively, decreasing the number of embedding changes to embed the same payload will, in general, decrease the embedding impact and thus lead to a more

secure scheme. However, practical steganographic schemes must have usable steganographic capacity. While a robust, 1-bit watermark may be very useful for applications, a steganographic scheme that can communicate only 1 bit in an image is not practical. Thus, the primary goal of new steganographic algorithms is to develop statistically undetectable methods with high steganographic capacity.

An important concept in steganography is *embedding efficiency*, which is defined as the number of secret message bits embedded per unit distortion. If the impact of all embedding modifications is approximately the same, we can measure embedding efficiency as the number of message bits embedded per one embedding change. This concept is amenable to analysis using coding theory and has produced some interesting results that will be discussed in Section 12.5.1.

2.5.4 Blind or Informed Extraction

The concept of blind or informed message extraction is absent from the steganographic literature. It is usually implicitly assumed that the recipient of the covert message does not have the original cover Work available for use in decoding. In practice, however, this need not be the case. For example, both Alice and Bob may have the same database of images that they agree to use as cover Works. In this case, the use of informed extraction algorithms may be beneficial as the embedder will not need to embed the covert signal as strongly. Consequently, the risk of an adversary detecting the covert communication will be smaller.

2.5.5 Blind or Targeted Steganalysis

All steganalysis algorithms can be categorized as either system attacks, targeted, or blind. System attacks use a fault in the implementation or weaknesses due to an insufficient stego keyspace. A steganalysis algorithm is described as blind if the method of detection is independent of the steganography method used. Conversely, a targeted steganalysis method is one that is designed to detect stego Works that are created by one or more particular steganographic methods (e.g., LSB embedding).

In blind methods, the Work is usually represented in some high-dimensional feature space. Machine learning methods are then used to distinguish between the clusters of cover and stego Works in the feature space. An important advantage of blind methods is that they can potentially detect an unknown stego scheme. They are also capable of classifying stego Works to individual steganographic methods.

Because targeted steganalytic methods need to be designed for each individual steganographic method, their construction cannot be automated. On the other hand, such methods may be more accurate than blind schemes.

2.5.6 Statistical Undetectability

The primary goal of steganography is to hide the very fact that communication is taking place. Thus, the stego Work containing the covert message should be inconspicuous to an adversary. This property is very elusive and is not easily formalized.

Modern steganography and steganalysis is concerned with those properties of the stego Work that can be quantified. Histograms and a variety of higher-order statistics are usually computed by steganalysts to evaluate whether a given Work is compliant with the intended use of the communication channel. The presence of statistical anomalies is assumed to be indicative of nonlegitimate use and may therefore be used by the adversary to decide that a covert communication is present or to at least examine the stego Work in more detail. An object that is statistically compliant with the channel will be judged as not containing a covert message.

It is important to realize that while this view of steganographic security permits quantification and precise mathematical formulation, it is only an approximation of the steganographic/steganalysis problem. For example, imagine that Alice and Bob communicate via email and embed their secret messages in digital images. Their communication is inspected by an automatic steganalysis algorithm that checks the statistical properties of the images for the presence of anomalies typically left by steganographic systems. Alice and Bob can enjoy great peace of mind if they simply type their messages on a piece of paper, then take a picture of the paper, compress the picture using JPEG, and attach it to their innocent emails. The images are unmodified in any way and thus should pass the statistical tests performed during steganalysis. Of course, this is an example of overt communication that would not pass through any warden who can read!

Despite these limitations, statistical undetectability is a widely used concept in steganalysis. This framework also assumes Kerckhoffs' Principle that the adversary has complete knowledge of the steganographic algorithm being used, except for a secret key. Kerckhoffs' Principle is imported from the cryptographic community. However, in steganalysis we seek to detect the presence of a covert message, not to determine the content of this message. The covert content can be protected by cryptographic means and attacked using cryptanalysis methods. However, this provides no protection against detection of the presence of a covert message. And detecting the presence of a covert message is often considerably easier than decoding its content.

A further assumption made by the statistical undetectability framework is that the adversary has complete knowledge of the statistical distribution of all valid cover Works. This knowledge provides the context within which stego Works are exchanged and against which any statistical anomalies are measured. At an abstract level, we can describe the statistics of the cover Works as a probability distribution on the space of all possible cover Works [60]. This statistical

model partially captures contextual (semantic) information. For example, if the communication channel is an Internet newsgroup discussing cars, a picture of a dragster will have a higher probability of occurrence than a picture of blueberry pudding. Taking the example from the beginning of this section, if Alice and Bob exchange images of written text, this will also shape their communication channel in a way that may become incompatible with the probability density function of "typical images." However, because it is inherently difficult to assign such probabilities to images, in practice, a simplified model of the cover Works is accepted. For example, pixels in digital images may be modeled as realizations of a Markov variable with a certain probability transition matrix. We then talk of statistical undetectability *with respect to a certain model*.

2.5.7 **False Alarm Rate**

In digital watermarking it is common to discuss the false positive rate, i.e., the probabilty that a watermark will be detected when, in fact, no watermark is actually present. A very similar concept is present in steganalysis. Here, the false positive or false alarm rate is the probability that a steganalysis algorithm will report the presence of a covert message when, in fact, none is present. The cost of a false alarm varies depending on the application. If the warden decides to block the transmission from Alice to Bob, then this may alert the two conspirators that they are under surveillance. Alternatively, the warden may choose to send all suspect stego Works to be investigated by humans or much more expensive computational analysis. If the false alarm rate is too high, these resources may quickly become overwhelmed.

It is easy to reduce the false positive rate to zero by simply never detecting the presence of a stego Work. However, then the probability of missing a stego Work, i.e., not recognizing that a Work contains a covert message, is 100%. Statistical undetectability is therefore most appropriately evaluated using receiver operating characteristic (ROC) curves (see Chapter 7), in which the tradeoff between false positives and false negatives is apparent.

2.5.8 **Robustness**

In watermarking, robustness measures the ability to detect the watermark after common signal processing operations. In steganography, the property is seldom, if ever, considered. This is because almost all steganography is assumed to occur over a digital network (e.g., the Internet), over which there are no degradations. Thus, what Alice transmits is exactly what Bob receives. Of course, if analog radio or television broadcasts were being used as cover Works for covert messaging, then robustness to the signal degradations common to these communications channels would have to be considered. However, for the remainder of this book, we assume that robustness is not an issue for the steganographic applications under consideration.

2.5.9 Security

In watermarking, security measures the ability to resist hostile attacks. In steganography, there are three types of warden: passive, active, and malicious. Most current steganographic methods are designed for the passive warden scenario in which the warden just passively observes the communication and does not interfere with the Work in any way. An active warden may introduce distortion whose goal is to prevent steganographic communication. For example, a Work might be compressed or downsampled to fit within a given bandwidth. The third type, the malicious warden, may intentionally try to remove the hidden message or impersonate the communicating parties. Steganography with an active warden has received much less attention in the literature. The interested reader is directed to Craver [99].

Most publications on steganography commonly use the term security as an equivalent of undetectability. In other words, a secure steganographic scheme means a statistically undetectable scheme.

2.5.10 Stego Key

A stego key is used in conjuction with a publicly known steganographic algorithm to embed a covert message into a cover Work. As in digital watermarking, the key may control, for example, the locations where the cover Work is modified (the embedding path) or may be used to generate other primitives that enter the embedding process (e.g., the noise sequences in the segment on stochastic modulation in Section 12.4.3).

The covert message may be cryptographically encrypted prior to embedding, in which case, there is also a crypto key. While the stego and crypto keys could be derived from one master key, it is simpler to consider the two keys separately.

The cryptographic scheme may be symmetric or nonsymmetric (public key cryptography). In the latter case, we speak of public key steganography [22].

The size of the stego keyspace is an important attribute that may compromise an otherwise secure steganographic scheme. Imagine, for example, that there are only 2^{32} possible stego keys. We could detect (and, in fact, even extract) the secret message by trying every possible key and stop once a meaningful message is obtained. This could be automated by simply calculating the entropy of the extracted bitstream and/or inspecting header information for the purpose of recognizing known file formats. If the embedded bitstream is encrypted, this attack would not work. However, we may compare statistical properties of pixels along the alledged embedding path with the statistics of all pixels in the image. If the correct path is followed, then the number of embedding modifications will be higher than when following a random path through the image [151]. This can be formalized using hypothesis testing.

A stego keyspace, like a watermark keyspace, is very different from a crypto keyspace. This is because in steganalysis, we strive to detect the presence of

the message rather than its content. For example, imagine that the stego key controls the pseudo-random path followed by the embedder. The reliability of detection of LSB embedding using methods described in Section 13.2.2, such as the Sample Pairs Analysis, is independent of the length of the stego key! What influences warden's success rate when using this attack is the number of embedding changes, not the key length.

2.6 EVALUATING AND TESTING STEGANOGRAPHIC SYSTEMS

How do we evaluate the vast number of steganographic schemes available on the Internet today (*http://www.stegoarchive.com*)? Is it possible to decide if one steganographic scheme is better than other schemes? Are there performance measures or standard tests that one could use for this purpose? Unfortunately there are currently no universally accepted tests or standards that can give us quick answers to these questions. On the other hand, there exist guidelines and general attack philosophies that should be considered when designing or selecting a steganographic method.

There are a number of blind steganalysis algorithms that can be used to test any steganographic method. To do so requires a large and diverse database of test Works that, ideally, would accurately model the conditions under which the steganographic algorithm is to be used. Such a database is divided into two sets, the training set and the test set. The training set contains a set of original cover Works together with a set of stego Works, both of which are used to train the blind steganalysis classifier. The previously unseen (to the steganalysis algorithm) test set, which also contains original cover Works and stego Works, can then be used to measure the false positive and false negative rates of the steganalysis algorithm for the particular steganographic method under evaluation. While such a procedure provides some assurances as to the security of the steganalysis algorithm, it must be remembered that this is an empirical procedure that is dependent on both the steganalysis algorithm and the test database. Changing one or both of these may give very different results.

An encouraging evaluation using the best current blind steganalysis methods still does not mean that a steganographic scheme is undetectable. There still might exist targeted schemes that utilize a specific feature of the embedding scheme, as we will see in Chapter 13.

The reliability of steganalysis methods is known to be very sensitive to the origin and preprocessing of the cover Works. This is especially true for hiding in the pixel domain. For example, a simple resizing of an image may increase the number of unique colors in the image and confuse some steganalytic methods [382]. In this book, tests of steganographic schemes are carried out for three different databases of images described in Section 13.1.3.

Thus, testing should be performed using many steganalysis tools and many databases. For a "practically secure" steganographic scheme, there should not exist a successful targeted attack and it should resist steganalysis using current blind steganalysis engines. As in cryptography, the concept of practical security is tied to our inability to attack the scheme. The status of a steganographic method may suddenly change with the appearance of a new steganalysis method. The best a designer of a steganographic scheme can do is to follow certain design principles, avoid known pitfalls, and incorporate certain mathematical constructs, as discussed in Chapter 12, that are guaranteed to decrease the statistical undetectability of the embedding changes.

2.7 SUMMARY

This chapter has discussed several possible applications of watermarking and steganography, and how to judge their suitability for various systems.

For watermarking, the following are the main points covered.

- Most applications of watermarks can be performed with other technologies. However, watermarks offer three potential advantages over other techniques:
  - They are imperceptible.
  - They do not get removed when Works are displayed or converted to other file formats.
  - They undergo the same transformations as the Works in which they are embedded.
- We described the following eight possible applications in detail:
  - *Broadcast monitoring:* Identifying when and where Works are broadcast by recognizing watermarks embedded in them.
  - *Owner identification:* Embedding the identity of a Work's copyright holder as a watermark.
  - *Proof of ownership:* Using watermarks to provide evidence in ownership disputes.
  - *Transaction tracking:* Using watermarks to identify people who obtain content legally but illegally redistribute it.
  - *Content authentication:* Embedding signature information in content that can be later checked to verify it has not been tampered with.
  - *Copy control:* Using watermarks to tell recording equipment what content may not be recorded.

- *Device control:* Using watermarks to make devices, such as toys, react to displayed content.
- *Legacy enhancements:* Using watermarks to improve the functionality of existing (legacy) systems.

- The suitability of a given watermarking system for a given application may be judged in terms of the following properties of that system:
  - *Embedding effectiveness:* The probability that the embedder will successfully embed a watermark in a randomly selected Work.
  - *Fidelity:* The perceptual quality of watermarked content.
  - *Data payload:* The amount of information that can be carried in a watermark.
  - *Blind or informed detection:* Whether (a) the watermark detector can detect a watermark in a Work without extra information (*blind detection*), or (b) the detector requires some information related to the original version of a watermarked Work (*informed detection*).
  - *False positive rate:* The frequency with which we should expect watermarks to be falsely detected in unwatermarked content.
  - *Robustness:* The ability of the watermark to survive normal processing of content.
  - *Security:* The ability of the watermark to resist hostile attacks.
  - *Cipher and watermark keys:* Whether the system employs (a) cipher keys to control message encryption and (b) watermark keys to control embedding and detection.
  - *Modification and multiple watermarks:* The possibility of changing embedded watermarks or embedding several watermarks in one Work.
  - *Cost:* The computational cost of the embedder and detector.
- The required properties of a watermark strongly depend on the application. Thus, the appropriate evaluation criteria are application dependent.
- Benchmarking is a reasonable means of comparing watermarking systems. However, no benchmark is likely to be relevant to all applications.
- Watermarking systems should be tested on large datasets.
- If a watermarking system is improved so that it has better performance in one property, this can often be traded for better performance in other properties.

For steganography, the following are the main points covered.

- Steganography is a privacy tool.
- Steganography is broken if the mere existence of a secret message is detected.
- Message undetectability can usually be guaranteed by making sure the message is sufficiently short.
- Steganography systems should be tested using targeted and blind attacks. Tests should be made on large and diverse datasets.
- Some applications of steganography can be performed with other technologies. However, steganography offers the advantage that even the knowledge that Alice and Bob are communicating is kept secret.
- We described the following two possible applications in detail:
  - Covert communications by dissidents.
  - Covert communications by criminals.
- We discussed a number of properties of steganographic systems and indicated which properties of watermarking are *not* relevant to steganography.
- *Embedding effectiveness*—not relevant.
- *Fidelity*—not necessarily relevant.
- *Steganographic capacity* is equivalent to a watermark's payload. It is the maximum number of bits that can be hidden in a given cover Work, such that the probability of detection by an adversary is negligible.
- *Embedding capacity* is the maximum number of bits that can be hidden in a given cover Work.
- *Embedding efficiency* is the number of secret message bits embedded per unit distortion.
- *Blind or informed extraction* refers to whether the recipient also has a copy of the original cover Works.
- *System steganalysis* refers to attacks on stego systems that use an implementation weakness rather than the nature of embedding modifications.
- *Blind steganalysis* refers to detection methods that are independent of the steganography method used.
- *Targeted steganalysis* refers to detection methods that are created specifically for the detection of one or more particular steganographic methods (e.g., LSB embedding).

- *Statistical undetectability* is the probability of detecting a stego Work based on assumed distributions of cover and stego Works.
- *False alarm rate* is the probability that a steganalysis algorithm will report the presence of a covert message when none is present.
- *Robustness* is seldom discussed in steganography. This is because almost all steganography is assumed to occur over a digital network (e.g., the Internet), over which there are no degradations.
- *Security* refers to the situation in which the warden is active or malicious, rather than passive.
- *Stego key* is used in conjuction with a publicly known steganographic algorithm to embed a covert message into a cover Work.

This page intentionally left blank

CHAPTER 3

Models of Watermarking

The first two chapters of this book have discussed the field of watermarking and steganography at a fairly nontechnical level. With the present chapter, we begin our exploration of the detailed, technical principles. In this chapter, and throughout the rest of the book, we illustrate these principles with algorithms and experiments described in special sections called *Investigations*.

The goal of this chapter is to familiarize the reader with several conceptual models of watermarking. These models serve as ways of thinking about actual watermarking systems. They fall into two broad groups: models based on a view of watermarking as a method of communications, and models based on geometric views of watermarking algorithms.

The chapter begins with some background material. Section 3.1 describes the basic notational conventions used in equations throughout the book. Section 3.2 gives a brief description of traditional communications systems, outside the context of watermarking. The next two sections form the bulk of the chapter. In Section 3.3, we describe three different ways of modeling watermarking in terms of traditional communications. This is followed, in Section 3.4, by a discussion of geometric models of watermarking.

Finally, we conclude the chapter with a brief discussion of *correlation-based watermarking*. This is a subclass of possible watermarking algorithms that we believe encompasses the majority of proposed systems. Much of the analysis presented in the rest of this book is focused on these types of systems.

A note on terminology. In the foregoing chapters, we have been using the term *watermark* rather loosely. There are, however, several subtly different objects to which this term might refer: a pattern added to a Work, a message encoded by that pattern, or a pattern used for comparison during watermark detection. In informal discussions, the distinctions between these objects are not very important. However, for more technical discussions, we must keep them clear. We therefore give each of them a unique name: *added pattern, message,* and *reference pattern,* respectively (see Section 3.3.1). We continue to use the term *watermark* informally to mean any of these things.

3.1 NOTATION

In this section, we define some preliminary notational conventions. The section does not describe all notation used in this book; rather, it defines the notation necessary to begin our technical discourse. As we proceed through this chapter, and through the rest of the book, we introduce additional notation as needed. A complete list of notational conventions and commonly used variables is found in Appendix D. The following is organized according to the various types of variables we use.

- *Scalars:* Scalar values and individual members of sets are set in italic, upper- or lowercase: N, i, x, and so on. The magnitude of scalar value n is denoted $|n|$.
- *Sets:* Sets are set in a calligraphic font. For example, the set of real numbers is $\mathcal{R}$, and a set of messages is $\mathcal{M}$. The number of elements in set $\mathcal{M}$ is denoted $|\mathcal{M}|$.
- *Vectors and Arrays:* One-dimensional vectors and higher-dimensional arrays (both of which are usually referred to as vectors in this text) are represented as boldface, lowercase letters; as, for example, $\mathbf{c}$, $\mathbf{v}$, and $\mathbf{w}$. Indices into these arrays are specified in square brackets. For example, the pixel of image $\mathbf{c}$ at location x, y is specified by $\mathbf{c}[x, y]$.

  We are sometimes interested in a frequency transform domain representation of a vector. Typical frequency transforms discussed are the Fourier transform and the discrete cosine transform. In either case, we refer to the transform of the vector $\mathbf{c}$ with uppercase ($\mathbf{C}$) and rely on the context of the discussion to indicate the implied transform.

  We often use subscripts to indicate different versions of vectors. For example, if we know that a Work, $\mathbf{c}$, is an original Work, we indicate that fact by writing it as $\mathbf{c_o}$. A watermarked version of the Work is $\mathbf{c_w}$. In general, a Work that has been subjected to a noise process is $\mathbf{c_n}$, and a noisy, watermarked Work is denoted $\mathbf{c_{wn}}$. Occasionally, subscripts are also used to index into arrays of vectors.

  The Euclidian length of vector $\mathbf{c}$ is denoted $|\mathbf{c}|$.

  The *sample mean* of a vector, denoted $\overline{\mathbf{c}}$ for vector $\mathbf{c}$, is the average of its elements. The *sample variance,* $s^2_{\mathbf{c}}$, is the average of $(\mathbf{c}[i] - \overline{\mathbf{c}})^2$.
- *Random Scalar Variables:* Random scalar variables are indicated with a sans serif italic font: r, m, and so on. Each such variable is associated with a probability distribution or density function. The probability that the value r will be drawn from the distribution of r is written $P_r(r)$.

  The *statistical mean* of r (i.e., the expected value of r) is written as μ_r. The *statistical variance* of r (i.e., the expected value of $(r - \mu_r)^2$) is written as σ^2_r.
- *Random Vectors:* Random vectors are in the same font as random scalars, but are bold: $\mathbf{c}$, $\mathbf{w}$, and so on. As with random scalar variables, the

probability that vector **c** will be drawn from the distribution of **c** is written $P_{\mathbf{c}}(\mathbf{c})$.

The notation $\mu_{\mathbf{c}}$ indicates the statistical mean of the distributions from which the elements of **c** are drawn. Similarly, σ_c^2 is the statistical variance of those elements.

The sample mean of a random vector is itself a random value, $\bar{c}$. The probability that $\bar{c}$ takes on a given value is just the probability of drawing a vector from **c** that has a sample mean of that value. Similarly, the sample variance for a random vector is itself a random value, $s_{\mathbf{c}}$. Note that we are not always strict in denoting what is and is not a random variable, but instead make the distinction when the distinction is important.

3.2 COMMUNICATIONS

To understand the similarities and differences between watermarking and conventional communications, it is useful to briefly review the traditional model of a communications system. We first highlight some of the components of a communications system that will be relevant in our extension to watermarking. We then discuss a number of classes of transmission channels that have been found to be useful for thinking about watermarking. This section ends with a discussion of secure transmission systems.

3.2.1 Components of Communications Systems

Figure 3.1 illustrates the basic elements of the traditional communications model. We begin with a message, m, that we want to transmit across a communications channel. This message is encoded by the *channel encoder* in preparation for transmission over the channel. The channel encoder is a function that maps each possible message into a code word drawn from a set of signals that can be transmitted over the channel. This code word is conventionally denoted as **x**.

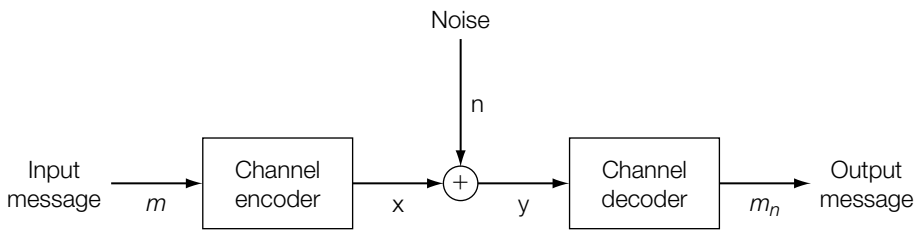


FIGURE 3.1

Standard model of a communications system.

In practice for digital signals, the encoder is usually broken down into a *source coder* and a *modulator*. The source coder maps a message into a sequence of *symbols* drawn from some *alphabet*. The modulator converts a sequence of symbols into a physical signal that can travel over the channel. For example, it might use its input to modulate the amplitude, frequency, or phase of a physical carrier signal for radio transmission.

The actual form of the channel encoder's output depends on the type of transmission channel being used, but for our purposes we will think of $\mathbf{x}$ as a sequence of real values, $\mathbf{x} = \{\mathbf{x}[1], \mathbf{x}[2], \ldots, \mathbf{x}[N]\}$, quantized to some arbitrarily high precision. We also assume that the range of possible signals is limited in some way, usually by a power constraint that says that

$$\sum_i (\mathbf{x}[i])^2 \leq p, \tag{3.1}$$

where p is a constant limit on power.

The signal, $\mathbf{x}$, is then sent over the transmission channel, which we assume is noisy. This means that the received signal, conventionally denoted $\mathbf{y}$, will generally be different from $\mathbf{x}$. The change from $\mathbf{x}$ to $\mathbf{y}$ is illustrated here as resulting from *additive noise*. In other words, the transmission channel is thought of as adding a random noise signal, $\mathbf{n}$, to $\mathbf{x}$.

At the receiving end of the channel, the received signal, $\mathbf{y}$, enters into the *channel decoder,* which inverts the encoding process and attempts to correct for transmission errors. This is a function that maps transmitted signals into messages, m_n. The decoder is typically a many-to-one function, so that even noisy code words are correctly decoded. If the channel code is well matched to the given channel, the probability that the decoded message contains an error is negligibly small.

3.2.2 Classes of Transmission Channels

In designing a communications system of the type pictured in Figure 3.1, we usually regard the transmission channel as fixed. That is, we cannot design or modify the noise function that occurs during transmission. The channel can be characterized by means of a conditional probability distribution, $P_{\mathbf{y}|\mathbf{x}}(\mathbf{y})$, which gives the probability of obtaining $\mathbf{y}$ as the received signal if $\mathbf{x}$ is the transmitted signal.

Different transmission channels can be classified according to the type of noise function they apply to the signal and how that noise is applied. As previously mentioned, the channel illustrated in Figure 3.1 is an *additive noise* channel, in which signals are modified by the addition of noise signals, $\mathbf{y} = \mathbf{x} + \mathbf{n}$. The noise signals might be drawn from some distribution independent of the signal being modified. The simplest channel to analyze (and perhaps the most important) is a Gaussian channel, in which each element of the noise

signal, $\mathbf{n}[i]$, is drawn independently from a Normal distribution with zero mean and some variance, $\sigma_{\mathbf{n}}^2$.

However, there are several nonadditive types of channels that will be of concern. One of the most important is a *fading channel,* which causes the signal strength to vary over time. This can be expressed as a scaling of the signal, $\mathbf{y} = \nu[t]\mathbf{x}$, where $0 \leq \nu[t] \leq 1$ is an unknown value that may vary slowly with time or with each use of the channel. Such a channel might also include an additive noise component, rendering $\mathbf{y} = \nu[t]\mathbf{x} + \mathbf{n}$.

3.2.3 Secure Transmission

In addition to modeling the characteristics of the channel, we must often be concerned with the access an adversary may have to that channel. In particular, we are interested in applications that demand security against both *passive* and *active* adversaries. In the former case, an adversary passively monitors the transmission channel and attempts to illicitly read the message. In the latter case, the adversary actively tries to either disable our communications or transmit unauthorized messages. Both forms of adversary are common in military communications. A passive adversary simply monitors all military communications, whereas an active adversary might attempt to jam communications on the battlefield. These security threats were briefly discussed in Sections 2.3.7 and 2.3.8, and two approaches were outlined to defend against attacks: cryptography and spread spectrum communications.

Prior to transmission, cryptography is used to encrypt a message, or *cleartext,* using a key. It is this encrypted message (or *ciphertext*) that is transmitted. At the receiver, the ciphertext is received and then decrypted using the same or a related key to reveal the cleartext. This arrangement is depicted in Figure 3.2.

There are two uses of cryptography. The first is to prevent passive attacks in the form of unauthorized reading of the message. The second is to prevent active attacks in the form of unauthorized writing. However, cryptography

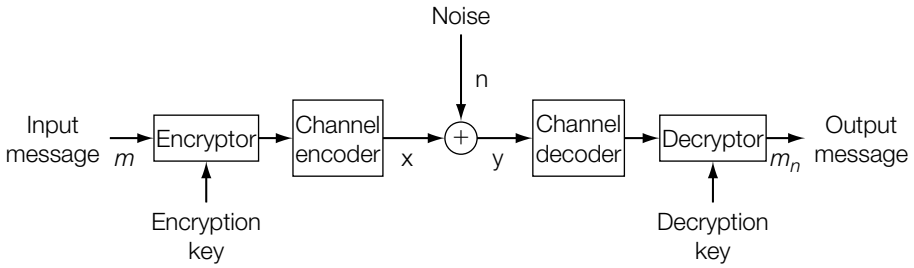


FIGURE 3.2

Standard model of a communications channel with encryption.

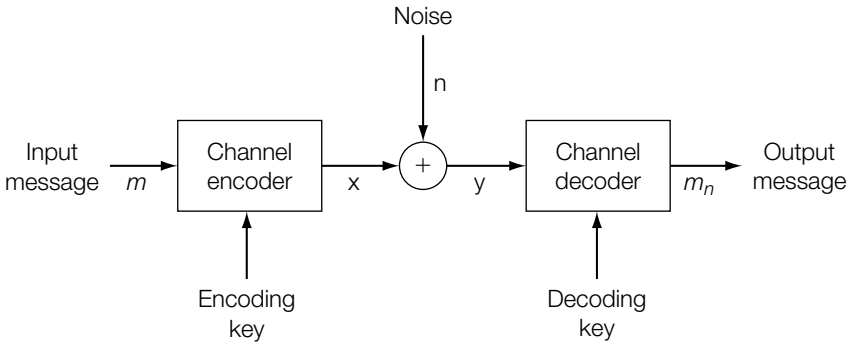


FIGURE 3.3

Standard model of a communications channel with key-based channel coding.

does not necessarily prevent an adversary from knowing that a message is being transmitted. In addition, cryptography provides no protection against an adversary intent on jamming or removing a message before it can be delivered to the receiver.

Signal jamming (i.e., the deliberate effort by an adversary to inhibit communication between two or more people) is of great concern in military communications and has led to the development of *spread spectrum communication*. Here, modulation is performed according to a secret code, which spreads the signal across a wider bandwidth than would normally be required [332]. The code can be thought of as a form of key used in the channel coder and decoder, as illustrated in Figure 3.3.

The idea of spread spectrum communication is best illustrated with an example. One of the earliest and simplest spread spectrum technologies is known as *frequency hopping*[1] [283]. As the name suggests, in a frequency-hopping system the transmitter broadcasts a message by first transmitting a fraction of the message on one frequency, the next portion on another frequency, and so on. The pattern of hops from one to another frequency is controlled by a key that must be known to the receiver as well as the transmitter. Without this key, an adversary cannot monitor the transmission. Jamming the transmission is also very difficult, in that it could only be done by introducing noise at *all* possible frequencies, which would require too much power to be practical.

[1] Frequency hopping was invented by Hedy Kiesler Markey and George Antheil. They are an unlikely pair to have invented such a major idea in communications. Hedy Kiesler Markey was a famous movie and pinup star in the 1930s and 1940s, better known by her stage name, Hedy Lamarr. George Antheil was a pianist with whom Hedy Lamarr was rehearsing when she had the idea of frequency hopping. His main contribution was to suggest that player-piano rolls be used to encode the sequence of frequencies [53].

Spread spectrum communications and cryptography are complementary. Spread spectrum guarantees delivery of signals. Cryptography guarantees secrecy of messages. It is thus common for both technologies to be used together. For those familiar with layered models of communications, spread spectrum can be thought of as responsible for the *transport layer,* and cryptography as responsible for the *messaging layer*.

3.3 COMMUNICATION-BASED MODELS OF WATERMARKING

Watermarking is, in essence, a form of communication. We wish to communicate a message from the watermark embedder to the watermark receiver. It is natural, then, to try to fit watermarking into the traditional model of a communications system.

In this section, we look at three ways to do this. The differences between these models lie in how they incorporate the cover Work into the traditional communications model. In the basic model, presented first, the cover Work is considered purely as noise. In the second model, the cover Work is still considered noise, but this noise is provided to the channel encoder as *side information*. Finally, the third model does not consider the cover Work as noise, but rather as a second message that must be transmitted along with the watermark message in a form of multiplexing.

3.3.1 Basic Model

In Figures 3.4 and 3.5, we show one way in which watermarking can be mapped into the framework of Figure 3.3. Figure 3.4 shows a system that uses an informed detector, and Figure 3.5 shows a system that uses a blind detector (these terms are defined in Chapter 2). In this mapping, watermarking is viewed as a transmission channel through which the watermark message is communicated. The cover Work is part of that channel.

Regardless of whether we are using an informed detector or a blind detector, the embedding process consists of two basic steps. First, the message is mapped into an *added pattern,* $\mathbf{w_a}$, of the same type and dimension as the cover Work, $\mathbf{c_o}$. For example, if we are watermarking images, the watermark encoder would produce a two-dimensional pixel pattern the same size as the cover image. When watermarking audio, the watermark encoder produces an audio signal. This mapping might be done with a watermark key.

In many of the examples of embedding algorithms presented in this book, the added pattern is computed in several steps. We generally begin with one or more *reference patterns,* $\mathbf{w_{r0}}$, $\mathbf{w_{r1}}$, ..., which are predefined patterns, possibly dependent on a key. These are combined to produce a pattern that encodes

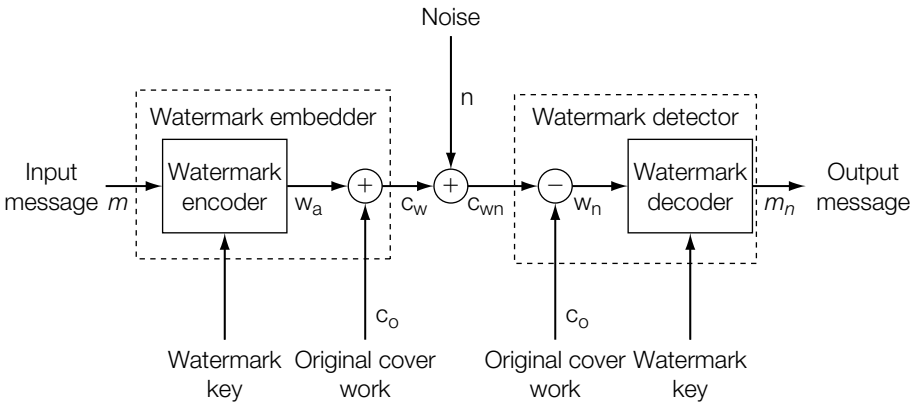


FIGURE 3.4

Watermarking system with a simple informed detector mapped into a communications model.

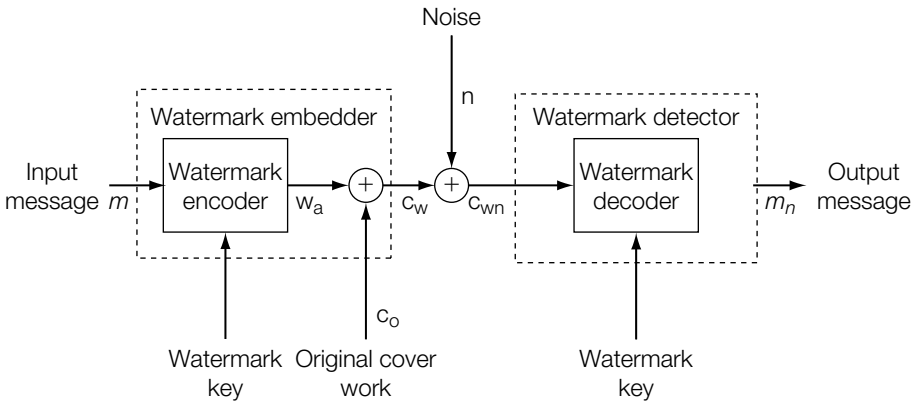


FIGURE 3.5

Watermarking system with blind detector mapped into a communications model. (Note that in this figure there is no meaningful distinction between the watermark detector and the watermark decoder.)

the message, which we refer to as a *message pattern,* $\mathbf{w}_m$. The message pattern is then scaled or otherwise modified to yield the added pattern, $\mathbf{w_a}$.

Next, $\mathbf{w_a}$ is added to the cover Work, $\mathbf{c_o}$, to produce the watermarked Work, $\mathbf{c_w}$. We refer to this type of embedder as a *blind embedder* because the encoder ignores the cover Work (in contrast to an *informed embedder,* defined in Section 3.3.2).

After the added pattern is embedded, we assume that the watermarked Work, $\mathbf{c_w}$, is processed in some way, and we model the effect of this processing as the addition of noise. The types of processing the Work might go through include compression and decompression, broadcast over analog channels, image or audio enhancements, and so on. Such processing might also include malicious practices by adversaries intent on removing the watermark. The former distortions are discussed in Chapter 9, and security against tampering is discussed in Chapter 10. Note that all these types of processing are dependent on the watermarked Work; therefore, it is a simplification here to model their effects with additive noise.

If we are using an informed watermark detector (Figure 3.4), the detection process can consist of two steps. First, the unwatermarked cover Work may be subtracted from the received Work, $\mathbf{c_{wn}}$, to obtain a received noisy watermark pattern, $\mathbf{w_n}$. This is then decoded by a *watermark decoder,* with a watermark key. Because the addition of the cover Work in the embedder is exactly cancelled out by its subtraction in the detector, the only difference between $\mathbf{w_a}$ and $\mathbf{w_n}$ is caused by the noise process. Therefore, we can ignore the addition of the cover Work, which means that the watermark encoder, the noise process, and the watermark decoder together form a system that is exactly analogous to the communications system shown in Figure 3.3.

In more advanced informed detection systems, the entire unwatermarked cover Work is not necessary. Instead, some function of $\mathbf{c_o}$, typically a data-reducing function, is used by the detector to cancel out "noise" effects represented by the addition of the cover Work in the embedder. For example, in Cox *et al*. [96], only a small fraction of DCT coefficients from the original image are required at the detector.

In a blind watermark detector (Figure 3.5), the unwatermarked cover Work is unknown, and therefore cannot be removed prior to decoding. Under these circumstances, we can make the analogy with Figure 3.3, where the added pattern is corrupted by the addition of a single noise pattern formed by the combination of the cover Work and the noise signal. The received, watermarked Work, $\mathbf{c_{wn}}$, is now viewed as a corrupted version of the added pattern, $\mathbf{w_a}$, and the entire watermark detector is viewed as the channel decoder.

In applications that require robustness, such as transaction tracking or copy control, we would like to maximize the likelihood that the detected message is identical to the embedded one. This is the same as the objective in a traditional communications system. However, it should be noted that in authentication applications the goal is not to communicate a message but to learn whether and how a Work has been modified since a watermark was embedded. For this reason, the models in Figures 3.4 and 3.5 are not typically used to study authentication systems.

Using the model of Figure 3.5, we can now construct a simple example of an image watermarking system with a blind detector.

INVESTIGATION

Blind Embedding and Linear Correlation Detection

The embedding algorithm in the system we describe here implements a blind embedder. We denote this algorithm by E_BLIND, which refers to this *specific* example of blind embedding rather than the generic concept of blind embedding. In fact, there are many other algorithms for blind embedding.

The detection algorithm uses linear correlation as its detection metric. This is a very common detection metric, which is discussed further in Section 3.5.

To keep things simple, we code only one bit of information. Thus, m is either 1 or 0. We assume that we are working with only grayscale images. Most of the algorithms presented in this book share these simplifications. Methods of encoding more than one bit are discussed in Chapters 4 and 5.

System 1: E_BLIND/D_LC

This system employs a single reference pattern, $\mathbf{w_r}$, which is an array of pixel intensities the same size as the cover image. This pattern might be predefined or generated randomly based on a watermark key. The message pattern, $\mathbf{w}_m$ (which encodes our 1-bit message, m), is equal to either $\mathbf{w_r}$ or $-\mathbf{w_r}$, depending on whether $m = 1$ or $m = 0$, respectively.

In the embedder, the message pattern, $\mathbf{w}_m$, is scaled by an input parameter, α, to yield the added pattern. The value α controls the tradeoff between visibility and robustness of the watermark. Thus, the blind embedding algorithm computes the following:

$$\mathbf{w}_m = \begin{cases} \mathbf{w_r} \text{ if } m = 1 \\ -\mathbf{w_r} \text{ if } m = 0 \end{cases} \tag{3.2}$$

$$\mathbf{w_a} = \alpha \mathbf{w}_m \tag{3.3}$$

$$\mathbf{c_w} = \mathbf{c_o} + \mathbf{w_a}. \tag{3.4}$$

To detect the watermark, we must detect the signal $\mathbf{w_r}$ in the presence of the noise caused by $\mathbf{c_o}$ and $\mathbf{n}$. As is discussed in Section 3.5, the optimal way of detecting this signal in the presence of additive Gaussian noise is to compute the linear correlation between the received image, $\mathbf{c}$, and the reference pattern, $\mathbf{w_r}$, as

$$z_{\text{lc}}(\mathbf{c}, \mathbf{w_r}) = \frac{1}{N}\mathbf{c} \cdot \mathbf{w_r} = \frac{1}{N}\sum_{x,y} \mathbf{c}[x, y]\, \mathbf{w_r}[x, y], \tag{3.5}$$

where $\mathbf{c}[x, y]$ and $\mathbf{w_r}[x, y]$ denote the pixel values at location x, y in $\mathbf{c}$ and $\mathbf{w_r}$, respectively, and N is the number of pixels in the image.

If $\mathbf{c} = \mathbf{c_o} + \mathbf{w_a} + \mathbf{n}$, then

$$z_{\text{lc}}(\mathbf{c}, \mathbf{w_r}) = \frac{1}{N}(\mathbf{c_o} \cdot \mathbf{w_r} + \mathbf{w_a} \cdot \mathbf{w_r} + \mathbf{n} \cdot \mathbf{w_r}). \tag{3.6}$$

Assuming that $\mathbf{c_o}$ and $\mathbf{n}$ are drawn from Gaussian distributions, $\mathbf{c_o} \cdot \mathbf{w_r}$ and $\mathbf{n} \cdot \mathbf{w_r}$ are almost certain to have small magnitudes. On the other hand, $\mathbf{w_a} \cdot \mathbf{w_r} = \pm\alpha \mathbf{w_r} \cdot \mathbf{w_r}$ should have a much larger magnitude. Therefore, $z_{lc}(\mathbf{c}, \mathbf{w_r}) \approx \alpha \mathbf{w_r} \cdot \mathbf{w_r}/N$ for Works watermarked with $m = 1$, and $z_{lc}(\mathbf{c}, \mathbf{w_r}) \approx -\alpha \mathbf{w_r} \cdot \mathbf{w_r}/N$ for Works watermarked with $m = 0$.

In most applications, we would like the detector[2] to output a distinct message if the received content probably contains no mark. If the detector receives $\mathbf{c} = \mathbf{c_o} + \mathbf{n}$, we should expect the magnitude of $z_{lc}(\mathbf{c}, \mathbf{w_r})$ to be very small. We can therefore determine whether a watermark is present by placing a threshold, τ_{lc}, on the magnitude of $z_{lc}(\text{boxrandbfc}, \mathbf{w_r})$. If $|z_{lc}(\mathbf{c}, \mathbf{w_r})| < \tau_{lc}$, then the detector reports that no watermark is present.

Thus, the D_LC detector outputs

$$m_n = \begin{cases} 1 & \text{if } z_{lc}(\mathbf{c}, \mathbf{w_r}) > \tau_{lc} \\ \textit{no watermark} & \text{if } -\tau_{lc} \le z_{lc}(\mathbf{c}, \mathbf{w_r}) \le \tau_{lc} \\ 0 & \text{if } z_{lc}(\mathbf{c}, \mathbf{w_r}) < -\tau_{lc}. \end{cases} \tag{3.7}$$

The detection threshold has a direct effect on the false positive rate. A lower threshold means that there is a higher chance that an unwatermarked Work will be classified as watermarked. Methods for estimating the false positive probability from the detection threshold are discussed in Chapter 7.

Experiments

To test this system, we must choose a reference pattern to embed. In our first experiment, we used a white noise pattern with each pixel value drawn at random from a uniform distribution. The pattern was normalized to have unit variance and the embedding strength, α, was set to 1.0.

The E_BLIND embedder was applied to each of 2,000 images under the two conditions $m = 1$ and $m = 0$, yielding a total of 4,000 watermarked images. The D_LC detector was then applied. Figure 3.6 shows the distributions of $z_{lc}(\mathbf{c_w}, \mathbf{w_r})$ obtained. The distribution of detection values resulting from 2,000 unwatermarked images is illustrated by a dotted line in Figure 3.6.

We can choose a detection threshold based on the resulting false positive probability. Selecting a detection threshold of $\tau_{lc} = 0.7$ yields an estimated false positive probability[3] of $P_{fp} \approx 10^{-4}$, according to a method introduced in Chapter 7. At this threshold, 57 images in which the message $m = 0$ was embedded were classified as having no watermark, as were 41 images in which the message $m = 1$ had been embedded. Recall from Chapter 2 that the *effectiveness* of a watermarking system is the probability that the output of the embedder will result

[2] We do not differentiate between a *watermark detector* and a *watermark decoder* here, or in most of the book.

[3] This false positive probability is too high for most applications. Nevertheless, it gives us a reference point to use in experiments.

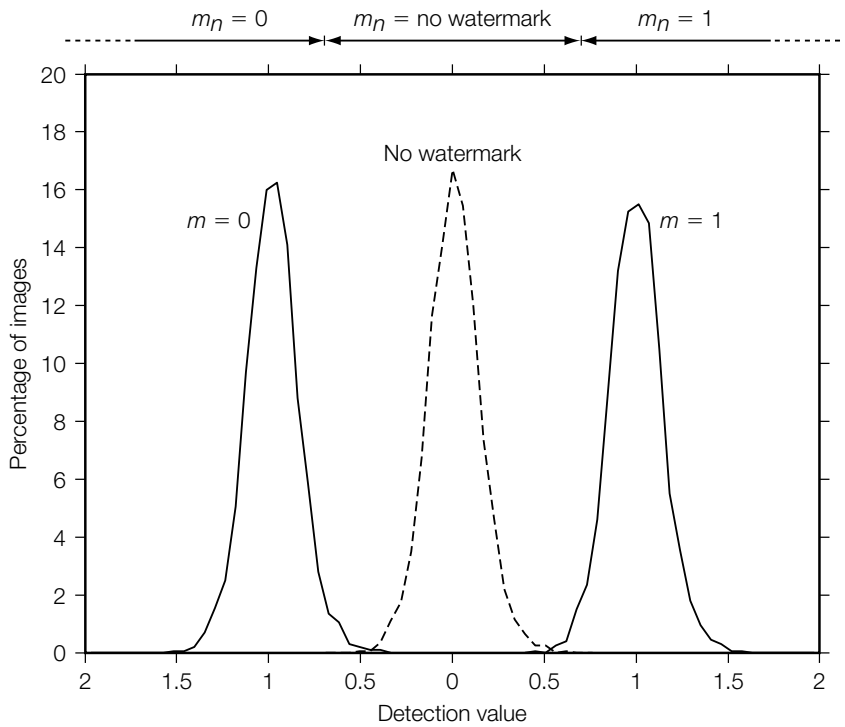


FIGURE 3.6

Distributions of linear correlation values resulting from System 1 (E_BLIND/D_LC) with a white noise reference pattern. The left-hand curve is the distribution when the embedded message was 0. The right-hand curve is the distribution when it was 1. The dashed curve is the distribution when no watermark was embedded. The legend at the top of the graph shows how the linear correlation decoder (D_LC) maps correlation values into messages.

in a positive, correct detection reported by the detector. Thus, with the current detection threshold, the measured effectiveness of the E_BLIND/D_LC system is approximately 98%.

The performance of this algorithm is highly dependent on the exact reference pattern used. For example, Figures 3.7 and 3.8 show the effects of using two perceptually different reference patterns with the same embedding strength. The reference pattern in Figure 3.8 was made using a pseudo-random number generator to select an independent value for each pixel. The reference pattern in Figure 3.8 was made by applying a low-pass filter to the reference pattern of Figure 3.7. Although the two reference patterns have been normalized to have the same variance, and the value of alpha is equal in both cases, the

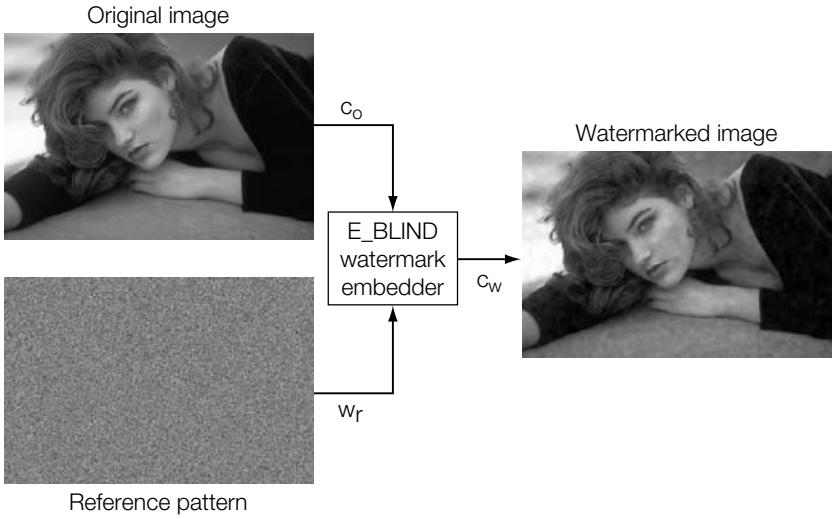


FIGURE 3.7

Results of the blind embedding algorithm, E_BLIND, with a reference pattern constructed from uniformly distributed noise.

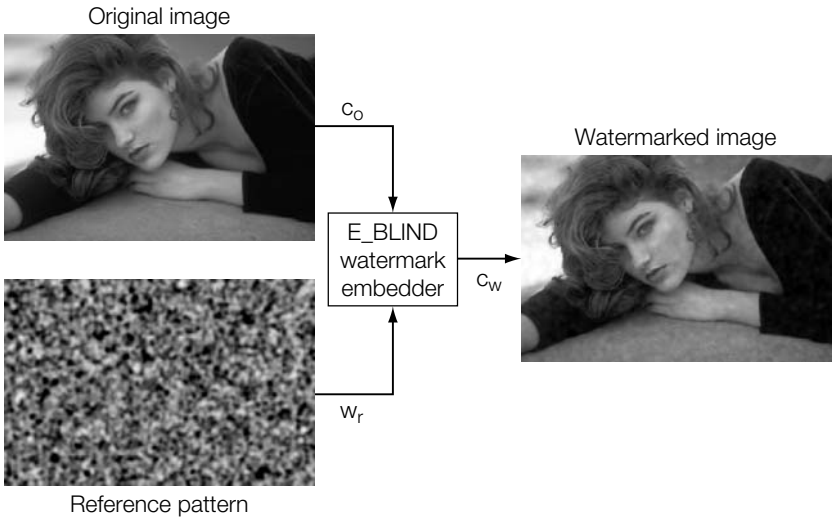


FIGURE 3.8

Results of the blind embedding algorithm, E_BLIND, with a reference pattern constructed from low-pass filtering the uniformly distributed noise of Figure 3.7.

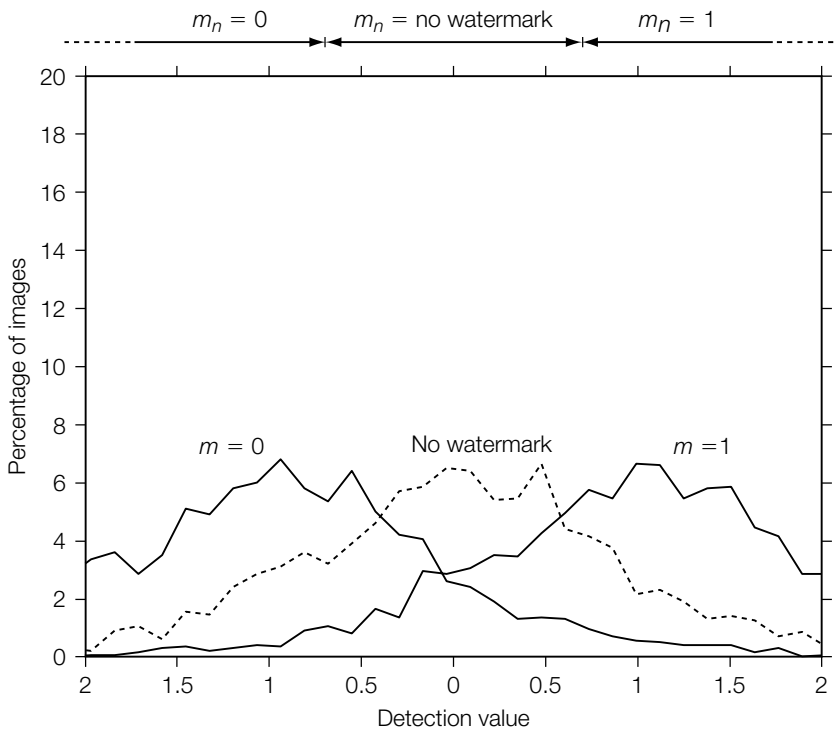


FIGURE 3.9

Distribution of linear correlations between images and a low-pass filtered random noise pattern.

watermarked image in Figure 3.8 has significantly worse fidelity than that in Figure 3.7, because the human eye is more sensitive to low-frequency patterns than to high-frequency patterns.

A second problem with low-frequency reference patterns is that they have a higher chance of generating false positives. This can be seen in a test we ran using the filtered pattern of Figure 3.7. The results of this second experiment are shown in Figure 3.9. The dashed curve shows the distribution of linear correlation values obtained when the detector is run on 2,000 unwatermarked images. The data corresponds to a measured false positive rate of 42% and can be compared to the dotted line in Figure 3.6, which was computed using the reference pattern from Figure 3.7. Clearly, the reference pattern in Figure 3.8 tends to have much higher magnitude correlations with unwatermarked images. This happens because images tend to have more energy in the low frequencies than in the high.

The high inherent correlations between the images and the reference pattern can also make it more difficult to embed a watermark. This can be seen by the distributions of detection values obtained from images watermarked with $m = 0$

and $m = 1$, shown in Figure 3.9. The measured effectiveness of the system using this low-frequency reference pattern is only about 68%.

We can mitigate these problems by restricting the set of patterns from which $\mathbf{w_r}$ is drawn to those similar to the one in Figure 3.7 (i.e., by using a pseudo-random number generator to independently decide on the value of each pixel). Each such pattern can be uniquely identified by the seed used for the pseudo-random number algorithm.

This watermarking system works and is actually suitable for some applications. However, linear correlation is optimal only when the image and the noise are drawn from Gaussian distributions. As discussed in Chapters 7 and 9, these assumptions are not often justified. Furthermore, as described in Chapter 10, this system is vulnerable to a number of attacks. Nevertheless, several proposed watermarking systems are very similar to this one (e.g., Bender *et al*. [35]), and it is useful as a starting point for our discussion.

3.3.2 Watermarking as Communications with Side Information at the Transmitter

Although some insights into robust watermarking with blind detectors can be gained by referring to the model in Figure 3.5, this model cannot accommodate all possible embedding algorithms, as it restricts the encoded watermark to be independent of the cover Work. Because the unwatermarked cover Work, $\mathbf{c_o}$, is obviously known to the embedder, there is no reason to enforce this restriction. Much more effective embedding algorithms can be made if we allow the watermark encoder to examine $\mathbf{c_o}$ before encoding the added pattern $\mathbf{w_a}$.

Figure 3.10 shows a model of watermarking that allows $\mathbf{w_a}$ to be dependent on $\mathbf{c_o}$. The model is almost identical to that in Figure 3.5, with the only difference being that $\mathbf{c_o}$ is provided as an additional input to the watermark encoder. Note that this change allows the embedder to set $\mathbf{c_w}$ to any desired value by simply letting $\mathbf{w_a} = \mathbf{c_w} - \mathbf{c_o}$. If we continue to consider the cover Work as part of the noise process ($\mathbf{c_o} + \mathbf{n}$) in the transmission channel, this new model is an example of a communications system with side information at the transmitter, as originally studied by Shannon [366]. In other words, the embedder is able to exploit some information about the channel noise, specifically $\mathbf{c_o}$ itself.

Since Shannon introduced the topic, several authors have studied communications with side information [88, 182, 456]. It has been discovered that in some types of channels it often does not matter whether the side information is available to the transmitter, the receiver, or both; its interference can be eliminated. Recently, a few researchers [78, 82, 95] have begun to apply

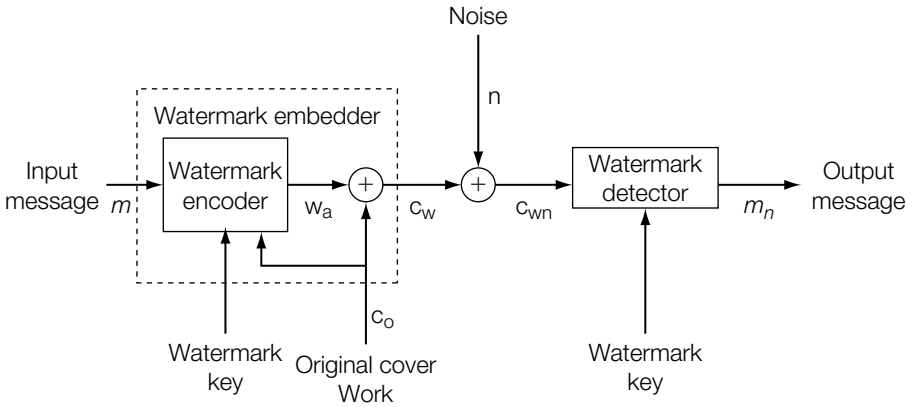


FIGURE 3.10

Watermarking as communications with side information at the transmitter.

the lessons learned about communications with side information to watermarking. We explore this topic in more detail in Chapter 5. For now, we illustrate some of the power of this approach by modifying the blind embedder/linear correlation detector (E_BLIND/D_LC) system to yield 100% effectiveness.

INVESTIGATION

Improving Effectiveness by Exploiting Side Information

In the preceding investigation, it was shown that the E_BLIND/D_LC watermarking system has less than 100% effectiveness when the detection threshold is $\tau_{lc} = 0.7$. This is evident in Figure 3.6, in that the tails of the distributions for $m = 1$ and $m = 0$ both cross the thresholds τ_{lc} and $-\tau_{lc}$. The portions of these distributions that lie between the thresholds represent images in which the system will fail to embed a watermark (i.e., the linear correlation detector will report m = *no watermark,* even if no noise has been added to the image after embedding). The images that have this problem are those that have an unusually high-magnitude correlation with the reference pattern. If $\mathbf{c_o} \cdot \mathbf{w_r}/N$ is too large, subtracting $\alpha\mathbf{w_r}$ from the image will fail to push the correlation below $-\tau_{lc}$, and we cannot embed $m = 0$. Similarly, if $\mathbf{c_o} \cdot \mathbf{w_r} / N$ is too far below zero, we cannot embed $m = 1$.

In applications that have strict requirements with respect to false positive rate, data payload, and fidelity, we may have to be satisfied with a system that has an effectiveness of less than 100%. However, there are other applications in which we can compromise some of these other properties in exchange for 100% effectiveness. For example, the tracking of movie dailies is an application that has

less stringent fidelity requirements. By monitoring the detection value *at the embedder,* we should be able to increase the embedding strength when necessary to ensure 100% effectiveness.

System 2: E_FIXED_LC/D_LC

The watermarking system used in this investigation employs a new embedding algorithm, designed to ensure 100% embedding effectiveness. This is done by adjusting the embedding strength, α, so that all watermarked images will have a fixed-magnitude linear correlation with the reference pattern. The embedding algorithm is thus called E_FIXED_LC. The system uses the same linear correlation detector described for System 1, D_LC.

To determine the necessary embedding strength, we first compute the inherent correlation between the cover Work and the message pattern, $\mathbf{c_o} \cdot \mathbf{w}_m / N$. In the E_BLIND embedder, we assumed that this value was small. In the E_FIXED_LC system, we do not make this assumption. Instead, we set α to explicitly account for the effects of this inherent correlation.

Our aim is to ensure that the magnitude of the detection strength is always some constant, greater than the detection threshold. We express this constant as $\tau_{lc} + \beta$, where τ_{lc} is the detection threshold, and $\beta > 0$ is an embedding strength parameter. The magnitude of the detection value obtained from an image immediately after embedding is given by

$$z_{lc}(\mathbf{c_w}, \mathbf{w}_m) = \frac{1}{N}(\mathbf{c_o} \cdot \mathbf{w}_m + \mathbf{w_a} \cdot \mathbf{w}_m), \tag{3.8}$$

where $\mathbf{w}_m$ is the message pattern and $\mathbf{w_a} = \alpha \mathbf{w}_m$. By substituting $\tau_{lc} + \beta$ for $z_{lc}(\mathbf{c_w}, \mathbf{w}_m)$, and solving for α, we obtain

$$\alpha = \frac{N(\tau_{lc} + \beta) - \mathbf{c_o} \cdot \mathbf{w}_m}{\mathbf{w}_m \cdot \mathbf{w}_m}. \tag{3.9}$$

After computing α with this formula, the rest of the E_FIXED_LC algorithm proceeds the same way as the E_BLIND algorithm.

Experiments

This system was applied twice to 2,000 images; once with $m = 0$ and once with $m = 1$. The embedding parameters were $\tau_{lc} = 0.7$ and $\beta = 0.3$. The resulting distributions of detection values, $z_{lc}(\mathbf{c_w}, \mathbf{w_r})$, are shown in Figure 3.11. For reference, the D_LC detector was also applied to the 2,000 unwatermarked images. The resulting distribution of detection values, labeled *no watermark,* is shown with a dashed line. Note that the unwatermarked distribution is identical to that shown in Figure 3.6, in that the two watermarking systems differ only in the embedding algorithm used, and the embedding algorithm has no effect on unwatermarked images.

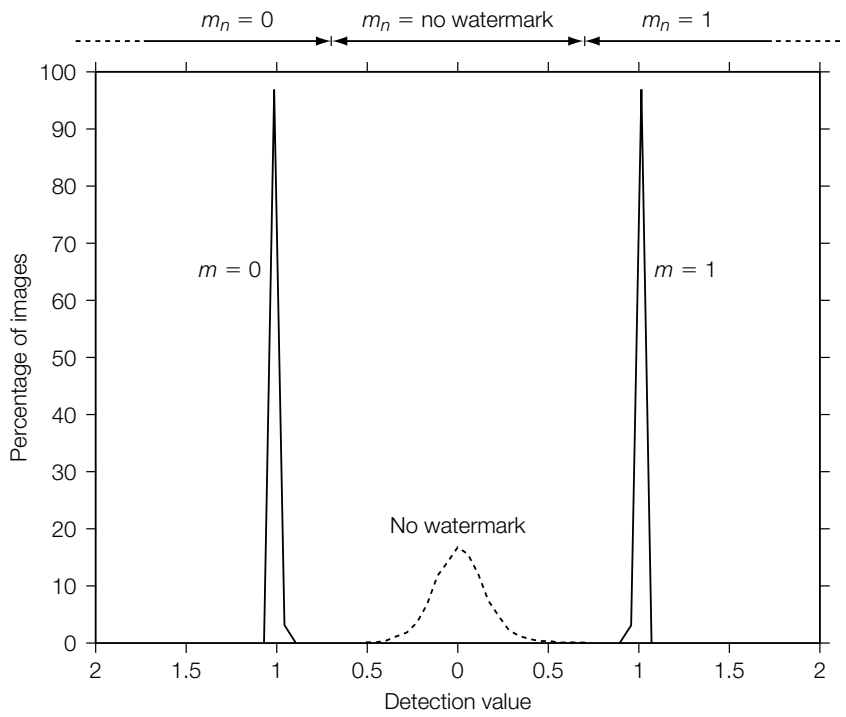


FIGURE 3.11

Distributions of linear correlation values resulting from the E_FIXED_LC/D_LC (System 2) watermarking system.

In theory, the detection values for all images embedded with $m = 1$ should be equal to 1.0, and, similarly, all $m = 0$ images should be equal to -1.0. However, because of round-off and truncation errors, there is some variation in the actual values obtained. Nevertheless, the distributions of detection values when a watermark is embedded are clearly much narrower than those resulting from the blind embedding algorithm (Figure 3.6).

With the detection threshold of $\tau_{lc} = 0.7$, all embedded watermarks were detected correctly. Thus, the system had a measured effectiveness of 100%.

3.3.3 Watermarking as Multiplexed Communications

Figure 3.12 shows an alternative model of watermarking as communications. Here, we no longer regard the cover Work as part of the transmission channel, but rather as a second message to be transmitted along with the watermark message in the same signal, $\mathbf{c_w}$. The two messages, $\mathbf{c_o}$ and m, will be detected

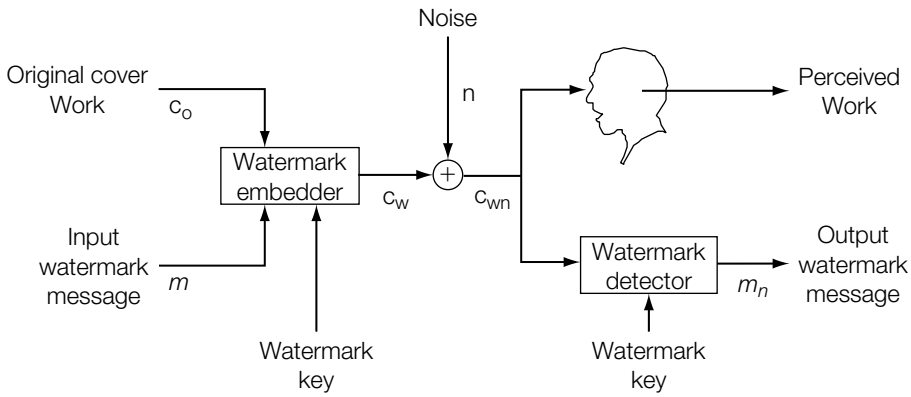


FIGURE 3.12

Watermarking as simultaneous communications of two messages. (Pictured with a blind watermark detector. An informed detector would receive the original cover Work as additional input.)

and decoded by two very different receivers: a human being and a watermark detector, respectively.[4]

The watermark embedder combines m and $\mathbf{c_o}$ into a single signal, $\mathbf{c_w}$. This combination is similar to the transmission of multiple messages over a single line in traditional communications by either time-division, frequency-division, or code-division multiplexing. One difference, however, is that in traditional communications the basic technology used for the different messages is the same, and the messages are separated by a single parameter (time, frequency, or code sequence). By contrast, in watermarking the two messages are separated by different technologies: watermark detection versus human perception. This is analogous to using, say, frequency division for one message and spread spectrum coding for the other.

After the signal passes through the transmission channel, it enters either a human perceptual system or a watermark detector. When viewing $\mathbf{c_{wn}}$, the human should perceive something close to the original cover Work, with no interference from the watermark. When detecting a watermark in $\mathbf{c_{wn}}$, the detector should obtain the original watermark message, with no interference from the cover Work. If the watermark detector is informed, it receives the original cover Work, or a function of the cover Work, as a second input.

This picture of watermarking emphasizes the symmetry between the watermark and the cover Work. One of the ways this symmetry manifests itself in the

[4] In some cases, watermark detection involves a human observer. For example, in Komatsu and Tominaga [239], a hidden image is made perceptible by rearranging the pixels of a cover image. A human observer is required to determine whether a valid hidden image is present.

watermarking literature is in two different usages of the term *signal-to-noise ratio* (SNR). In discussions of fidelity, "signal" refers to the cover Work and "noise" refers to the watermark. In discussions of effectiveness and robustness, it is the other way around: "signal" refers to the watermark and "noise" refers to the cover Work and/or any subsequent distortions. However, the former is usually referred to as the document-to-noise ratio (DWR), while the latter is termed the watermark-to-noise ratio (WNR). It is usually clear from the context which of these meanings is intended.

The symmetry in Figure 3.12 implies that solutions to problems in transmitting m should often have corresponding solutions to problems in transmitting $\mathbf{c_o}$. For example, in the informed embedding algorithm, E_FIXED_LC, we saw that we can solve the problem of effectiveness in blind embedding by examining the cover Work before designing the added pattern. The embedder computes the interference between the cover Work and the reference pattern, by computing $\mathbf{c_o} \cdot \mathbf{w_r}$, and adjusts the amplitude of the added pattern to compensate, by changing the value of α. Similarly, when faced with fidelity problems, we can improve the system by first using a perceptual model to examine how the watermark interferes with the Work, and then adjusting $\mathbf{w_a}$ to reduce this interference as much as possible. This idea is discussed in detail in Chapter 8.

3.4 GEOMETRIC MODELS OF WATERMARKING

The types of models described in the previous section allow us to draw from the field of communications when designing watermarking systems. Although we make use of such models in this book (primarily in Chapters 4 and 5), we also often conceptualize watermarking algorithms in *geometric* terms. This geometric view of watermarking is the topic of this section.

To view a watermarking system geometrically, we imagine a high-dimensional space in which each point corresponds to one Work. We refer to this space as *media space*. Alternatively, when analyzing more complicated algorithms, we may wish to consider projections or distortions of media space. We refer to such spaces as *marking spaces*. The system can then be viewed in terms of various regions and probability distributions in media or marking space. These include the following:

- The *distribution of unwatermarked Works* indicates how likely each Work is.
- The *region of acceptable fidelity* is a region in which all Works appear essentially identical to a given cover Work.
- The *detection region* describes the behavior of the detection algorithm.
- The *embedding distribution* or *embedding region* describes the effects of an embedding algorithm.

- The *distortion distribution* indicates how Works are likely to be distorted during normal usage.

We begin this section by discussing each of the previously listed distributions and regions in terms of media space. We then introduce the idea of marking space and discuss how it can be employed in designing watermarking systems. This is illustrated with a watermarking system (System 3 E_BLK_BLIND/D_BLK_CC), which is slightly more complicated than the two described so far.

3.4.1 Distributions and Regions in Media Space

Works can be thought of as points in an N-dimensional *media space*. The dimensionality of media space, N, is the number of samples used to represent each Work. In the case of monochrome images, this is simply the number of pixels. For images with three color planes (e.g., red, green, and blue), N is three times the number of pixels. For continuous, temporal content (such as audio or video), we assume that the watermark is embedded in a fixed-length segment of the signal and that the content is time sampled. Thus, for audio, N would be the number of samples in the segment. For video, N would be the number of frames in a segment multiplied by the number of pixels per frame (multiplied by 3 if the video is in color).

Because we are concerned here with digital content, each sample is quantized and bounded. For example, each pixel value of an 8-bit grayscale image is quantized to an integer between 0 and 255. This means that there is a finite (though huge) set of possible Works, and they are arranged in a rectilinear lattice in media space. Points between the lattice points, or outside the bounds, do not correspond to Works that can be represented in digital form. However, the quantization step size is generally small enough, and the bounds large enough, that we often gloss over this fact and assume that media space is continuous (i.e., that all points in the space, even those off the lattice, correspond to realizable Works). In the following we discuss each of the various probability distributions over media space, and regions within media space, that are of concern in analyzing watermarking systems.

Distribution of Unwatermarked Works

Different Works have different likelihoods of entering into a watermark embedder or detector. In audio, we are more likely to embed watermarks in music than in pure static. In video, we are more likely to embed watermarks in images of natural scenes than in video "snow." Content, such as music and natural images, possesses distinct statistical distributions [134, 354, 377, 431], and we must take these distributions into account.

When estimating the properties of a watermarking system, such as false positive rate and effectiveness, it is important to model the *a priori* distribution

of content we expect the system to process. This can be expressed either as a probability distribution over the lattice of points that represent digital Works or as a probability density function (PDF) over all points in media space.

There is a wide range of statistical models for the distribution of unwatermarked content. The simplest models assume that the distribution fits an elliptical Gaussian. Such a model is employed in the derivation of the E_BLIND/D_WHITE watermarking system (System 9) described in Chapter 7. More accurate models of most media can be obtained using Laplacian or generalized Gaussian distributions. These are used in the analysis of lossy compression in Chapter 9. The most sophisticated models try to describe content as the result of random, parametric processes. Such models are not used in this book, but the interested reader is referred to [206, 306, 414] for discussions of their application to images.

It is important to keep in mind that the distribution of unwatermarked content is application dependent. For example, satellite images are drawn from a very different distribution than news photographs. Music is drawn from a different distribution than speech. This variety of different distributions for different classes of Works can cause problems. If we estimate a watermark detector's false positive rate using one distribution, and then use the detector in an application in which the unwatermarked Works are drawn from a different distribution, our estimate will not be accurate. The problem can be very serious in applications that require extremely low false positive rates, such as copy control.

Region of Acceptable Fidelity

Imagine taking an original image, $\mathbf{c_o}$, and altering just a single pixel by one unit of brightness. This new image defines a new vector in media space, yet it is perceptually indistinguishable from the original image. Clearly, there are many such images, and we can imagine a region around $\mathbf{c_o}$ in which every vector corresponds to an image that is indistinguishable from $\mathbf{c_o}$. If $\mathbf{c_o}$ is an audio signal, similarly small changes will result in indistinguishable sounds, and they also form a region in media space. We refer to the region of media space vectors that are essentially indistinguishable from the cover Work, $\mathbf{c_o}$, as the *region of acceptable fidelity*.

It is extremely difficult to identify the true region of acceptable fidelity around a given Work, because too little is known about human perception. We usually approximate the region by placing a threshold on some measure of perceptual distance.[5] For example, it is common to use mean squared error (MSE) as a simple perceptual distance metric. This is defined as

[5] Note that by "measure of perceptual distance" we do not necessarily mean a perceptual distance *metric*. A true metric is symmetric and satisfies the triangle inequality [353]. Many of the functions in which we are interested are not symmetric. Therefore, we include the term *metric* only when discussing functions that are indeed metrics in the mathematical sense.

$$D_{\text{mse}}(\mathbf{c_1}, \mathbf{c_2}) = \frac{1}{N}\sum_i^N (\mathbf{c_1}[i] - \mathbf{c_2}[i])^2, \tag{3.10}$$

where $\mathbf{c_1}$ and $\mathbf{c_2}$ are N-dimensional vectors (N-vectors) in media space. If we set a limit, τ_{mse}, on this function, the region of acceptable fidelity becomes an N-dimensional ball (N-ball) of radius $\sqrt{N\tau_{\text{mse}}}$.

In practice, the MSE function is not very good at predicting the perceived differences between Works [208]. For example, if $\mathbf{c_1}$ is an image, and $\mathbf{c_2}$ is a version of $\mathbf{c_1}$ that has been shifted slightly to the left, the two Works will appear essentially identical, but the MSE might be enormous. In other words, MSE does not account for visual tracking. Another example can be seen in Figures 3.7 and 3.8. The MSE distances between the watermarked and unwatermarked images in these two figures are 16.4 and 16.3, respectively, yet the watermarked image in the latter figure is clearly far worse than that in the former.

Some perceptual distance functions are asymmetric. In these functions, the two arguments have slightly different interpretations. By convention, we interpret the first argument as an original Work and the second as a distorted version of it. For example, one commonly used asymmetric distance is based on the reciprocal of the SNR:

$$D_{\text{snr}}(\mathbf{c_1}, \mathbf{c_2}) = \frac{\sum_i^N (\mathbf{c_2}[i] - \mathbf{c_1}[i])^2}{\sum_i^N \mathbf{c_1}[i]^2}. \tag{3.11}$$

Here, the first argument, $\mathbf{c_1}$, is regarded as the signal, and the second, $\mathbf{c_2}$, as a noisy version of it. The distance measures how noisy $\mathbf{c_2}$ is in relation to $\mathbf{c_1}$.

More sophisticated distance functions that give better predictions of human judgment have been devised for both images and audio. These functions often measure perceptual distance in units of *just noticeable difference* (JND). Some general properties and uses of perceptual distance functions are discussed along with two specific functions in Chapter 8.

Detection Region

The *detection region* for a given message, m, and watermark key, k, is the set of Works in media space that will be decoded by the detector as containing that message. Like the region of acceptable fidelity, the detection region is often (but not always) defined by a threshold on a measure of the similarity between the detector's input and a pattern that encodes m. We refer to this measure of similarity as a *detection measure*.

In the D_LC detection algorithm, the detection measure is a linear correlation, $z_{\text{lc}}(\mathbf{c}, \mathbf{w_r})$. To find the shape of the detection regions for this detector, we begin by noting that the linear correlation between the received Work and the

reference pattern, $\mathbf{c} \cdot \mathbf{w_r} / N$, is equal to the product of their Euclidean lengths and the cosine of the angle between them, divided by N. Because the Euclidean length of $\mathbf{w_r}$ is constant, this measure is equivalent to finding the orthogonal projection of the N-vector $\mathbf{c}$ onto the N-vector $\mathbf{w_r}$. The set of all points for which this value is greater than τ_{lc} is just the set of all points on one side of a plane perpendicular to $\mathbf{w_r}$. The resulting half space is the detection region for $m = 1$. Similarly, the detection region for $m = 0$ is the set of all points on one side of a plane defined by $-\tau_{lc}$.

A successful embedding of a watermark message results in a watermarked Work that lies within the intersection of the region of acceptable fidelity and the detection region. Figure 3.13 illustrates this point, using a region of acceptable fidelity based on MSE and a detection region based on thresholding the linear correlation between the received Work and a reference pattern, $\mathbf{w_r}$ (as in the D_LC algorithm). The figure shows the two-dimensional slice of media space that contains the two vectors $\mathbf{c_o}$ and $\mathbf{w_r}$. The diagram is positioned so that $\mathbf{w_r}$ lies along the horizontal axis. Because randomly chosen vectors in high-dimensional space tend to be close to orthogonal, $\mathbf{c_o}$ lies near the vertical axis. The region of acceptable fidelity is an N-ball whose intersection with the diagram is a filled circle (or 2-ball). The plane that separates the detection region from the rest of media space intersects the diagram along a line perpendicular

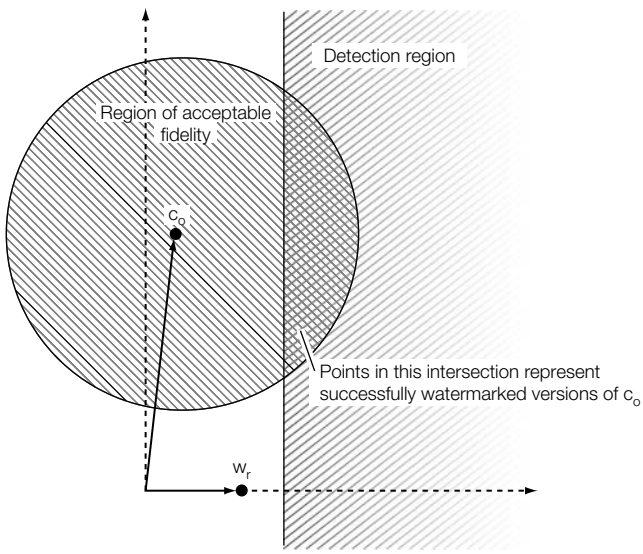


FIGURE 3.13

The region of acceptable fidelity (defined by MSE) and the detection region (defined by linear correlation) for a watermarking system such as the E_BLIND/D_LC system of System 1.

to $\mathbf{w_r}$. All points within the circular region of acceptable fidelity and to the right of the planar edge of the detection region correspond to versions of $\mathbf{c_o}$ that are within an acceptable range of fidelity and that will cause the detector to report the presence of the watermark. In other words, these points represent successful embeddings of the watermark message in the cover Work.

Embedding Distribution or Region

A watermark embedder is a function that maps a Work, a message, and possibly a key into a new Work. This is generally a deterministic function such that for any given original Work, $\mathbf{c_o}$, message, m, and key, k, the embedder always outputs the same watermarked Work, $\mathbf{c_w}$. However, the original Works are drawn randomly from the distribution of unwatermarked Works, and this leads us to view the embedder's output as random. The probability that a given Work, $\mathbf{c_w}$, will be output by the embedder is just the probability that a Work that leads to it, $\mathbf{c_o}$, is drawn from the distribution of unwatermarked Works. If several unwatermarked Works map into $\mathbf{c_w}$, the probability of $\mathbf{c_w}$ is just the sum of the probabilities of those Works. We refer to the resulting probability distribution as the *embedding distribution*.

Some embedding algorithms define an embedding distribution in which every point has a nonzero probability (assuming that every point in media space has a nonzero probability of entering the embedder). If we ignore the effects of clipping and rounding, this is the case with the embedding distribution defined by the E_BLIND image watermarking algorithm of System 1. Every possible image, even those outside the detection region, can be arrived at by applying the E_BLIND embedder to some other image. Figure 3.14 illustrates this point. Such algorithms necessarily have less than 100% effectiveness, in that there is some nonzero probability that the embedder will output a Work that is outside the detection region.

Other algorithms, such as the E_FIXED_LC algorithm of System 2, have a limited set of possible outputs. For a given reference pattern, message, and embedding strength, β, the E_FIXED_LC algorithm will only output images that lie on a fixed plane, as illustrated in Figure 3.15. With such a system, it makes sense to discuss an *embedding region* for a given message, which is the set of all possible outputs of the embedder. The system has 100% effectiveness if, and only if, the embedding region is completely contained within the detection region.

Distortion Distribution

To judge the effects of attacks on watermarked content, we need to know the probability of obtaining a given distorted Work, $\mathbf{c_{wn}}$, given that the undistorted, watermarked Work was $\mathbf{c_w}$. This conditional probability distribution is exactly the same type of distribution used to characterize transmission channels in

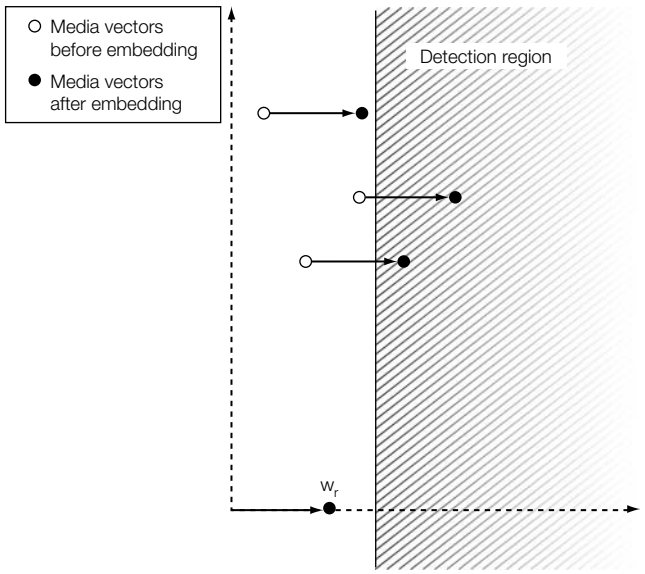


FIGURE 3.14

Effect of the E_BLIND embedding algorithm of System 1. The same vector, $\mathbf{w_a} = \alpha\mathbf{w_r}$, is added in every case, regardless of the original Work.

traditional communication theory. We refer to this as the *distortion distribution* around $\mathbf{c_w}$.

Theoretical discussions of watermarking often begin with the assumption that the distortion distribution can be modeled as additive Gaussian noise. This greatly simplifies analysis, but it does not model reality very well. Very few of the processes applied to digital content behave like Gaussian noise. In fact, very few of them are even random. During normal usage, the content most likely will be subject to distortions such as lossy compression, filtering, noise reduction, and temporal or geometric distortions. These are, in general, deterministic functions applied to the content. Therefore, the "noise" added as a result of the distortion is highly dependent on the content itself.

For example, consider the simple procedure of cropping a Work. In images, this is equivalent to setting columns and rows of pixels around the edges to black. In audio, it amounts to setting samples at the beginning or end of a clip to zero. These types of distortions can be expected to occur in many applications. Therefore, the result, $\mathbf{c_{wn}}$, must be assigned a nonzero probability in the distortion distribution around any Work. Note that $\mathbf{c_{wn}}$ can be quite far away from $\mathbf{c_w}$ in media space. Moreover, points between $\mathbf{c_{wn}}$ and $\mathbf{c_w}$ might be very unlikely to occur (we would not expect several columns of pixels near the right edge of an image to be reduced to half their brightness, while the other pixels remain untouched). This means that the distortion distribution

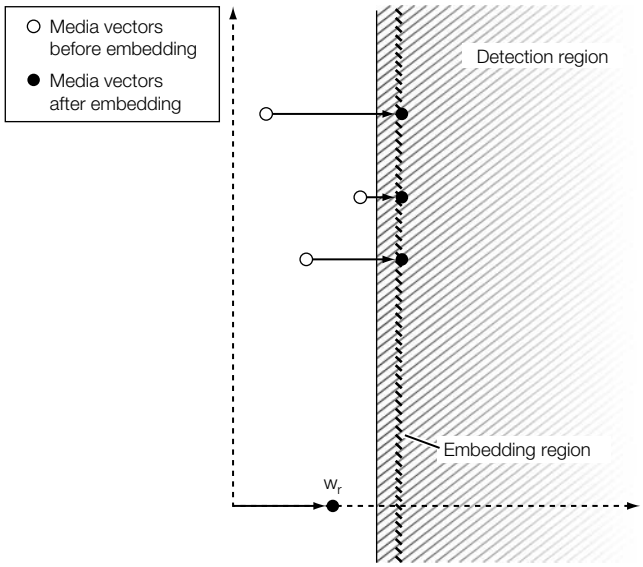


FIGURE 3.15

Effect of the E_FIXED_LC embedding algorithm of System 2. The vector added to each unwatermarked Work is chosen to guarantee that the resulting Work lies on a planar embedding region.

is multimodal, which is very unlike anything that might be produced by a Gaussian noise process. As we shall see in Chapter 7, the true nature of the distortion distribution has a dramatic effect on the optimality of certain detection measures.

3.4.2 Marking Spaces

For simple watermarking systems, such as the E_BLIND/D_LC and E_FIXED_LC/D_LC systems, identifying the embedding and detection regions in media space is not very difficult. However, most applications require more sophisticated algorithms, which are more difficult to analyze in media space. When considering such systems, it is often useful to view part of the system as performing a projection or distortion of media space into a *marking space*. The rest of the system can then be viewed as a simpler watermarking system, operating in this marking space rather than media space.

Watermark detectors are often designed with an explicit notion of marking space. Such a detector consists of a two-step process, as illustrated in Figure 3.16. The first step, *watermark extraction,* applies one or more preprocesses to the content, such as frequency transforms, filtering, block averaging, geometric or temporal registration, and feature extraction. The result is a

vector—a point in marking space—of possibly smaller dimensionality than the original. We refer to this vector as an *extracted mark*.

The second step is to determine whether the extracted mark contains a watermark and, if so, decode the embedded message. This usually entails comparing the extracted mark against one or more predefined *reference marks* (although other methods are possible, as is shown Chapter 5). This second step can be thought of as a simple watermark detector operating on vectors in marking space.

Watermark embedders are not usually designed with any explicit use of marking space, but they can be. Such an embedder would be a three-step process, as illustrated in Figure 3.17. The first step is identical to the extraction step in a watermark detector, mapping the unwatermarked Work into a point in marking space.

The second step is to choose a new vector in marking space that is close to the extracted mark and (hopefully) will be detected as containing the desired watermark. We refer to the difference between this new vector and the original, extracted mark as the *added mark*. This second step can be thought of as a simple watermark embedder operating in marking space.

The third step is to invert the extraction process, projecting the new vector back into media space to obtain the watermarked Work. Our aim here is to find

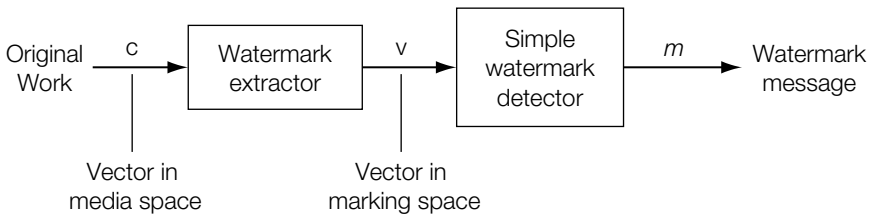


FIGURE 3.16

General two-step outline of a watermark detector.

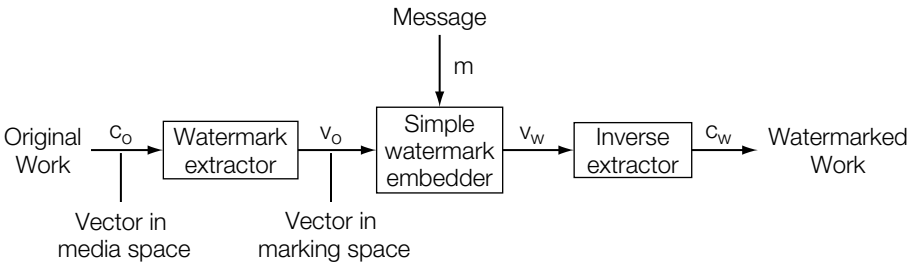


FIGURE 3.17

General three-step outline of a watermark embedder.

a Work that will yield the new vector as its extracted mark. If marking space has the same dimensionality as media space, this projection can be performed in a straightforward manner. However, if marking space has smaller dimensionality than media space, each point in marking space must correspond to many points in media space. Thus, there will be many Works that all yield the new vector as their extracted marks. Ideally, we would like to choose the one perceptually closest to the original Work. In practice, we generally use an approximate algorithm, which, while not giving us the absolute closest such Work, nevertheless leads to one that is reasonably close.

In systems designed according to Figures 3.16 and 3.17, one purpose of the extraction function is to reduce the cost of embedding and detection. A second purpose is to simplify the distribution of unwatermarked Works, the region of acceptable fidelity, and/or the distortion distribution so that simple watermarking algorithms will perform well. For example, by averaging groups of independent samples, we can obtain a marking space in which the distribution of unwatermarked Works is more closely Gaussian (because of the central limit theorem). By performing a frequency transform and scaling the terms by perceptually determined constants (see Chapter 8), we can obtain a marking space in which the region of acceptable fidelity is more closely spherical. By compensating for geometric and temporal distortions, we can obtain a marking space in which the distortion distribution is not multimodal.

To illustrate the idea of a marking space, we now develop our last watermarking system of the chapter. This system is built according to Figures 3.16 and 3.17.

INVESTIGATION

Block-Based, Blind Embedding, and Correlation Coefficient Detection

In the system investigated here, we extract watermarks by averaging 8×8 blocks of an image. This results in a 64-dimensional marking space. Watermarks are embedded in marking space by simple, blind embedding, and the resulting changes are projected back to the full size.

Watermarks are detected in marking space using the *correlation coefficient* as a detection measure. This is a normalized version of the linear correlation measure used in the D_LC detector, discussed in detail in Chapter 7. As in the previous algorithms, the embedded message is a single bit, m, and the detector reports either the value of the embedded bit or that no watermark is present.

System 3: E_BLK_BLIND/D_BLK_CC

We begin by describing the D_BLK_CC detection algorithm, and then proceed to describe the E_BLK_BLIND embedder. The detection algorithm operates in two steps, as illustrated in Figure 3.16, namely:

1. Extract a mark, **v**, from the received Work, **c**.

2. Use a simple detection algorithm to detect a watermark in the extracted mark.

To extract a watermark from an image, the image is divided into 8 × 8 blocks, and all blocks are averaged into one array of 64 values. Thus, the extracted mark, **v**, is

$$\mathbf{v}[i,j] = \mathcal{X}(\mathbf{c}) = \frac{1}{B}\sum_{x=0}^{w/8}\sum_{y=0}^{h/8}\mathbf{c}[8x+i, 8y+j], \tag{3.12}$$

where $0 \le i < 8$ and $0 \le j < 8$ are indices into the extracted mark, w and h are the width and height of the image, and B is the number of blocks in the image. This process is illustrated in Figure 3.18.

To detect a watermark in an extracted mark, the extracted mark is compared against a predefined reference mark. The reference mark in this system is an 8 × 8 array of values (i.e., a vector in marking space).

In principle, to compare the extracted mark with the reference mark, the detector could use linear correlation in the same manner as the D_LC detector. However, as we shall see, this would lead to a detection algorithm mathematically identical to D_LC. To make the system more interesting, we use a modification of linear correlation, called the *correlation coefficient,* which is more robust to certain distortions.

Correlation coefficients differ from linear correlations in two ways. First, we subtract the means of the two vectors before correlating them. Thus, the detection value is unaffected if a constant is added to all elements of either of the vectors. Second, we normalize the linear correlation by the magnitudes of the two vectors. Thus, the detection value is uneffected if all elements of either vector are multiplied by a constant. As a result, a system using the correlation coefficient is robust to changes in image brightness and contrast. The correlation coefficient is defined as

$$z_{cc}(\mathbf{v}, \mathbf{w_r}) = \frac{\tilde{\mathbf{v}} \cdot \tilde{\mathbf{w}}_\mathbf{r}}{\sqrt{(\tilde{\mathbf{v}} \cdot \tilde{\mathbf{v}})(\tilde{\mathbf{w}}_\mathbf{r} \cdot \tilde{\mathbf{w}}_\mathbf{r})}}, \tag{3.13}$$

where $\tilde{\mathbf{v}} = (\mathbf{v} - \bar{\mathbf{v}})$ and $\tilde{\mathbf{w}}_\mathbf{r} = (\mathbf{w_r} - \bar{\mathbf{w}}_\mathbf{r})$, $\bar{\mathbf{v}}$ and $\bar{\mathbf{w}}_\mathbf{r}$ are the mean values of $\mathbf{v}$ and $\mathbf{w_r}$ (respectively), and

$$\tilde{\mathbf{v}} \cdot \tilde{\mathbf{w}}_\mathbf{r} = \sum_{x=0}^{7}\sum_{y=0}^{7}\tilde{\mathbf{v}}[x,y]\,\tilde{\mathbf{w}}_\mathbf{r}[x,y]. \tag{3.14}$$

The correlation coefficient can be interpreted as the inner product of $\tilde{\mathbf{v}}$ and $\tilde{\mathbf{w}}_\mathbf{r}$ after each has been normalized to have unit magnitude. This is simply the cosine of the angle between the vectors and is bounded, thus:

$$-1 \le z_{cc}(\mathbf{v}, \mathbf{w_r}) \le 1. \tag{3.15}$$

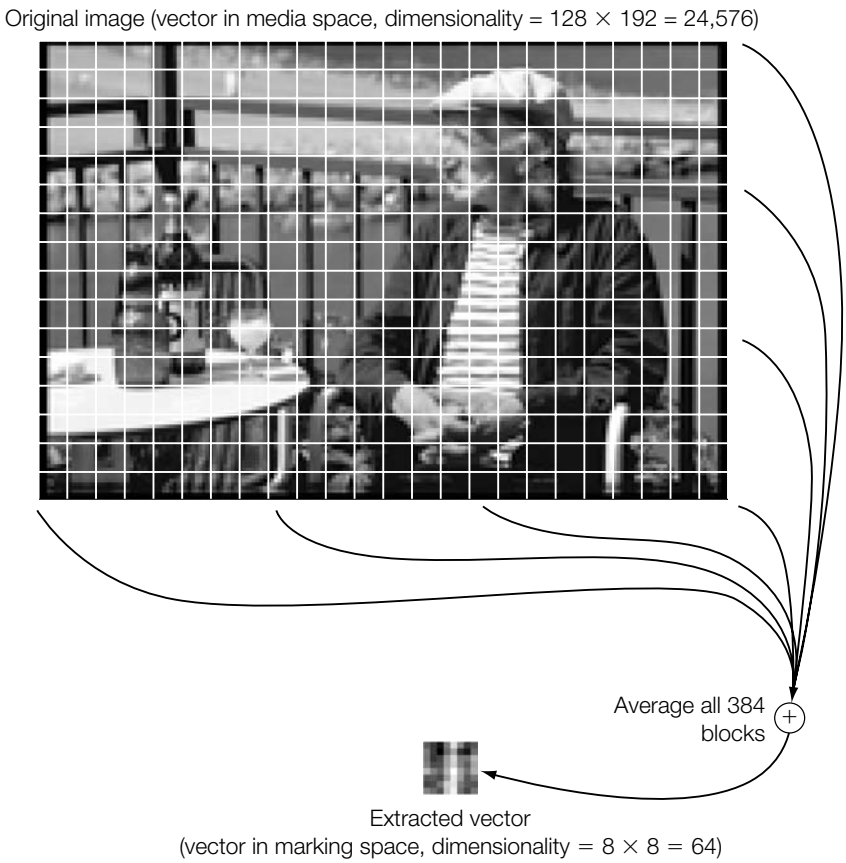

Original image (vector in media space, dimensionality = 128 × 192 = 24,576)
Average all 384 blocks
Extracted vector
(vector in marking space, dimensionality = 8 × 8 = 64)

FIGURE 3.18

Watermark extraction process for System 3, E_BLK_BLIND/D_BLK_CC (shown using a low-resolution image so that the blocks are large enough to illustrate).

Like the **D_LC** detector, the **D_BLK_CC** detector outputs

$$m_n = \begin{cases} 1 & \text{if } z_{cc}(\mathbf{v}, \mathbf{w_r}) > \tau_{cc} \\ \textit{no watermark} & \text{if } -\tau_{cc} \le z_{cc}(\mathbf{v}, \mathbf{w_r}) \le \tau_{cc} \\ 0 & \text{if } z_{cc}(\mathbf{v}, \mathbf{w_r}) < -\tau_{cc}, \end{cases} \tag{3.16}$$

where τ_{cc} is a constant threshold.

The embedding algorithm, **E_BLK_BLIND**, proceeds according to the three-step process previously described, namely:

1. Extract a mark, $\mathbf{v_o}$, from the unwatermarked Work, $\mathbf{c_o}$.
2. Choose a new vector, $\mathbf{v_w}$, in marking space that is close to the extracted mark but is (hopefully) inside the detection region.
3. Project the new vector back into media space to obtain the watermarked Work, $\mathbf{c_w}$.

The extraction process in the embedder is identical to that in the detector; namely, averaging 8 × 8 blocks to obtain a vector in 64-dimensional marking space.

For simplicity, watermarks are embedded in marking space using a blind embedding algorithm similar to E_BLIND. It might be preferable to use an embedder such as E_FIXED_LC, which yields a fixed detection value. However, constructing such an embedder for a correlation coefficient detector is rather complicated. We develop such embedders in Chapter 5. In the present system, the added mark, $\mathbf{w_a}$, is $\alpha\mathbf{w}_m$, where $\mathbf{w}_m = \mathbf{w_r}$ if $m = 1$, and $-\mathbf{w_r}$ if $m = 0$. The resulting vector is $\mathbf{v_w} = \mathbf{v_o} + \mathbf{w_a}$.

When projecting $\mathbf{v_w}$ into media space, we wish to find an image, $\mathbf{c_w}$, that is perceptually close to the original, $\mathbf{c_o}$. Because the extraction process is many-to-one, there are many possible ways to proceed. In this example we simply distribute each element of the desired change in the extracted mark uniformly to all contributing pixels. A simple way to do this is

$$\mathbf{c_w}[x, y] = \mathbf{c_o}[x, y] + (\mathbf{v_w}[x \bmod 8, y \bmod 8] - \mathbf{v_o}[x \bmod 8, y \bmod 8]), \quad (3.17)$$

where mod is the modulo operator. This step ensures that when the detector applies the extraction function to $\mathbf{c_w}$ the result will be $\mathbf{v_w}$, and the watermark will be detected.

Note that this embedding algorithm could be simplified by eliminating the previously listed Steps 1 and 2, because $\mathbf{v_w} - \mathbf{v_o}$ is just the added mark, $\mathbf{w_a}$, which can be determined without these steps. However, in later systems (presented in Chapter 5), we modify this system to use more sophisticated embedding methods for Step 2, and all three steps will be necessary. We therefore include all three steps here for purposes of illustration.

In principle, the performance of this watermarking system should be similar to that of the E_BLIND/D_LC system (System 1). In fact, if we were to use linear correlation in the detector, rather than the correlation coefficient, the performance would be *identical* to that of E_BLIND/D_LC using a reference pattern that consists of a single 8 × 8 pattern tiled over the full size of the image; that is, $\mathbf{w}'_\mathbf{r}[x, y] = \mathbf{w_r}[x \bmod 8, y \bmod 8]$ (see Figure 3.19). This follows from the fact that the entire algorithm is linear. If our detection statistic were $z_{\text{lc}}(\mathbf{v}, \mathbf{w_r}) = \mathbf{v} \cdot \mathbf{w_r} / 64$, then

$$z_{\text{lc}}(\mathbf{v}, \mathbf{w_r}) = \frac{1}{64} \sum_{i,j} \mathbf{v}[i, j]\, \mathbf{w_r}[i, j] \quad (3.18)$$

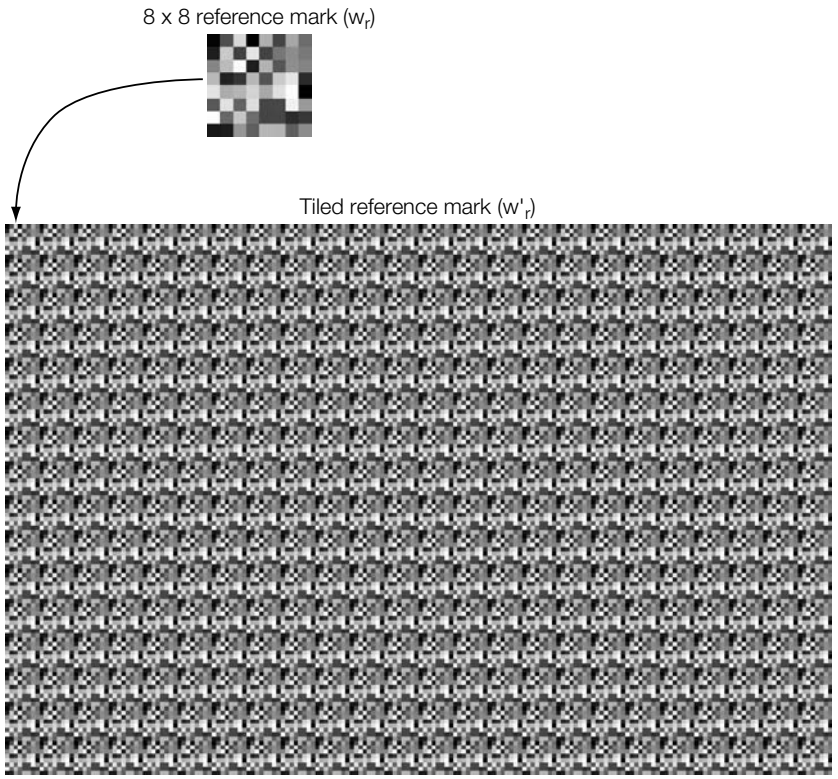


FIGURE 3.19

8 × 8 reference mark for the E_BLK_BLIND/D_BLK_CC watermarking system of System 3, and the same reference mark tiled to create a reference pattern for use in the E_BLIND/D_LC system.

$$= \frac{1}{64} \sum_{i,j} \left(\frac{1}{B} \sum_{x=0}^{w/8} \sum_{y=0}^{h/8} \mathbf{c}[8x + i,\, 8y + j] \right) \mathbf{w_r}[i,j] \tag{3.19}$$

$$= \frac{1}{64B} \sum_{x,y} \mathbf{w_r}[x \bmod 8,\, y \bmod 8]\mathbf{c}[x,y] \tag{3.20}$$

$$= \frac{1}{N} \sum_{x,y} \mathbf{w}'_{\mathbf{r}}[x,y]\mathbf{c}[x,y], \tag{3.21}$$

where N is the number of pixels in the image. Equation 3.21 is the same as the detection statistic used in D_LC. A similar argument shows that the E_BLK_BLIND embedder is essentially the same as the E_BLIND embedder. Thus, if we were using linear correlation as our detection measure, the block-averaging extraction process would have no effect. However, when we use the correlation coefficient for the detection measure, the extraction process has a substantial effect, in that it changes the factor by which the linear correlation is normalized.

This system has several strengths and weaknesses when compared to E_BLIND/D_LC. As mentioned, it is robust to certain changes in image brightness and contrast. The D_BLK_CC detector is also computationally cheaper than the D_LC detector. D_LC requires a multiply and add for each pixel when computing $\mathbf{c} \cdot \mathbf{w_r}$. In contrast, D_BLK_CC requires only one add per pixel when computing the extraction function, plus a handful of multiplies, adds, and divides when computing the correlation coefficient on the 8 × 8 extracted vector.

On the other hand, the number of possible reference marks, and hence the watermark keyspace, for E_BLK_BLIND/D_BLK_CC is much smaller than that for E_BLIND/D_LC. Moreover, many of those reference marks lead to poor statistical performance. This means that randomly generated reference marks tend not to work well, and the reference marks must be carefully selected.

Experiments

The E_BLK_BLIND/D_BLK_CC watermarking system was tested on 2,000 images using the reference mark illustrated in Figure 3.19. This mark was carefully selected to yield acceptable performance and was normalized to have unit

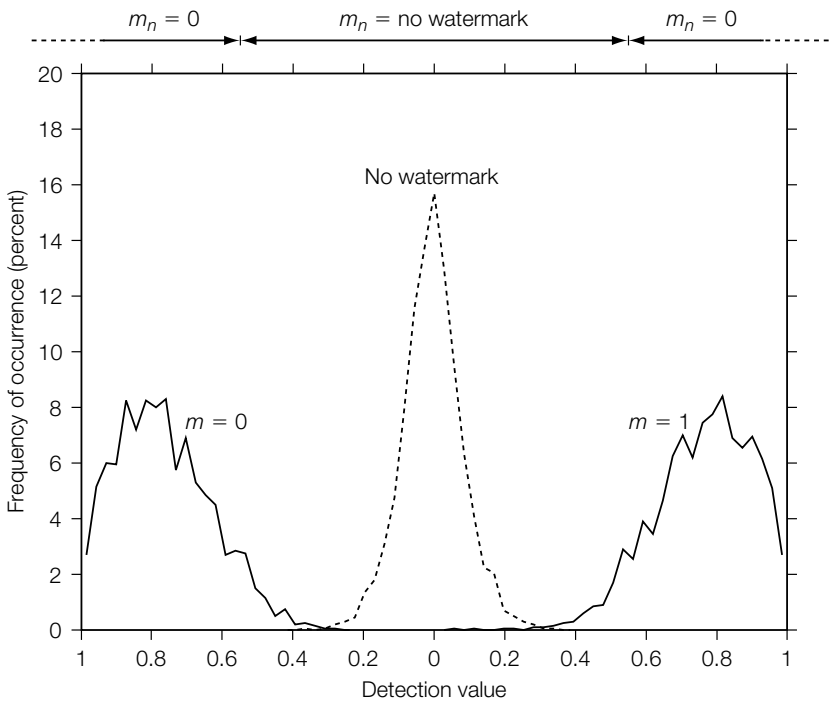


FIGURE 3.20

Performance of the block-based blind embedder with a correlation coefficient detector (E_BLK_BLIND/D_BLK_CC) of System 3 on 2,000 images.

variance. In Step 2 of the embedding algorithm, we set the embedding strength, $\alpha = 1$. The detection threshold was set to $\tau_{cc} = 0.55$, which gives an estimated false positive probability of about 10^{-6}, according to Equation 7.18 of Chapter 7.

The embedding performance is shown in Figure 3.20. As in Figures 3.6 and 3.11, this shows distributions of detection values for the images under each of $m = 1$, $m = 0$, and *no watermark* conditions. Note that the x-axis ranges from just -1 to 1, since the correlation coefficient is bounded. The system had a measured effectiveness of about 92%.

3.5 MODELING WATERMARK DETECTION BY CORRELATION

The preceding sections have discussed general models that may be used for virtually any watermarking system. The example systems we have used in our investigations, however, all fall into the class of *correlation-based* watermarking systems. Although this class does not include *all* possible watermarking systems, it does include the majority of the example systems presented in this book, and, we believe, the majority of systems proposed in the literature. We therefore conclude this chapter with a brief discussion of correlation-based detection measures.

There are several types of correlation used in watermark detection. The most basic is *linear correlation,* which we have seen in the D_LC detection algorithm. Other forms of correlation are distinguished by some normalization applied to vectors before the computation of their inner product. If we normalize two vectors to unit magnitude, we obtain the *normalized correlation* between them. Subtracting their means before computing normalized correlation gives us the *correlation coefficient* between them. We have seen the use of the correlation coefficient in the D_BLK_CC detection algorithm.

In some systems, such as those we have introduced so far, watermarking algorithms are explicitly built around the use of one of these types of correlation. In other systems, the use of correlation might be less obvious. Sometimes, a watermarking system is proposed that uses a detection method mathematically equivalent to the use of one form of correlation or another. These algorithms are often described without any reference to correlation.

Sometimes, correlation is a small part of the detection algorithm and is consequently not emphasized when the algorithm is described. For example, some detection algorithms proposed in the literature involve sophisticated feature detectors. The positions of detected features are either compared against a predefined set of positions [281] or are used to distort a watermark reference pattern that is then compared against the Work [32, 113, 452] (see Chapter 9). Although these comparisons are often performed using correlation, these algorithms are not typically considered correlation based, because their novelty and power lies in their use of feature detectors. Nevertheless, any analysis of their performance must involve an analysis of the effects of correlation.

The remainder of this section discusses the three main types of correlation measures previously described: linear correlation, normalized correlation, and the correlation coefficient. In each case, we provide some brief motivation for the measure's use, a geometric interpretation of the resulting detection region, and a discussion of equivalent detection methods.

3.5.1 Linear Correlation

The linear correlation between two vectors, $\mathbf{c}$ and $\mathbf{w_r}$, is the average product of their elements:

$$z_{\text{lc}}(\mathbf{v}, \mathbf{w_r}) = \frac{1}{N} \sum_i \mathbf{v}[i]\mathbf{w_r}[i]. \tag{3.22}$$

It is common practice in communications to test for the presence of a transmitted signal, $\mathbf{w_r}$, in a received signal, $\mathbf{v}$, by computing $z_{\text{lc}}(\mathbf{v}, \mathbf{w_r})$ and comparing it to a threshold. This practice is referred to as *matched filtering* and is known to be an optimal method of detecting signals in the presence of additive, white Gaussian noise.[6]

Geometric Interpretation

As pointed out in Section 3.4.1, the detection region that results from matched filtering comprises all points on one side of a hyperplane (see Figure 3.14). The hyperplane is perpendicular to the reference mark, and its distance from the origin is determined by the detection threshold.

One way of understanding the robustness of this detection region is to recognize that in high dimensions vectors drawn from a white Gaussian distribution tend to be nearly orthogonal to a given reference mark. Thus, when a vector is corrupted by additive white Gaussian noise, the noise tends to be parallel to the planar edge of the detection region, and rarely crosses that plane.

Equivalent Detection Methods

In general, any detection algorithm that computes a linear function of the samples in a Work and compares that function against a threshold can be considered to use linear correlation.

As a first example, an image watermark detection algorithm is proposed in Bender *et al.* [35], in which the pixels of an image are divided into two groups. The sum of the pixels in one group is subtracted from the sum of the pixels in

[6] Matched filtering is known to be optimal in the following sense: for any given probability of obtaining a false positive (deciding that $\mathbf{v} = \mathbf{w_r} + \mathbf{n}$ when, in fact, $\mathbf{v} = \mathbf{n}$), matched filtering minimizes the probability of obtaining a false negative (deciding that $\mathbf{v} = \mathbf{n}$ when, in fact, $\mathbf{v} = \mathbf{w_r} + \mathbf{n}$). This fact can be shown by showing that matched filtering is equivalent to *Neyman-Pearson hypothesis testing* [184].

the other group to obtain a detection statistic, which is then compared against a threshold to determine whether the watermark is present. This is equivalent to correlating the image against a pattern consisting of 1s and −1s. The pattern contains a 1 for each pixel in the group that is added into the detection statistic, and a −1 for each pixel that is subtracted from it.

A less obvious example, again from image watermarking, can be found in Koch and Zhao [237]. Here, a discrete cosine transform (DCT) is applied to each 8 × 8 block of the image, and selected coefficients are grouped as ordered pairs. The basic idea is to encode 1 bit of watermark information in each pair, according to whether or not the first coefficient in the pair is larger than the second. For example, if the first coefficient is larger, this might encode 1 bit. If it is smaller, this might encode 0 bit. To implement such an algorithm with linear correlation, we define a pattern for each bit (see Chapter 4 for more on this idea). All coefficients of the block DCT for each pattern are 0, except for the pair of coefficients used to encode the bit. The first coefficient in the pair contains a 1; the second contains a −1. Thus, if we correlate the block DCT of one of these patterns against the block DCT of the image, the sign of the result will tell us whether the first coefficient is larger than the second. Because the DCT is a linear transform, we can get the same result by correlating the pattern with the image directly in the spatial domain.[7]

Beyond the linear types of methods previously described, most nonlinear detection algorithms can be split into a nonlinear extraction method (see Section 3.4.2), followed by a correlation-based detector. This is a worthwhile exercise if the distributions in marking space that result from the extraction process are straightforward. For example, if the distribution of unwatermarked Works and the distortion distribution are approximately Gaussian, the performance of correlation-based detection in marking space is easy to analyze.

3.5.2 Normalized Correlation

One of the problems with linear correlation is that the detection values are highly dependent on the magnitudes of vectors extracted from Works. For many extraction methods, this means that the watermark will not be robust against such simple processes as changing the brightness of images or reducing the volume of music. It also means that even when reference marks are drawn from a white Gaussian distribution, a linear correlation detector's false positive probability is difficult to predict (see Chapter 7).

[7] The algorithms described here are not actually identical to that proposed in Koch and Zhao [237], because that system compares the *magnitudes* of the coefficients, rather than their signed values. This is equivalent to computing their correlation against a pattern dependent on the underlying Work. If both coefficients in a pair are positive, the pattern is constructed with +1 and −1 in the DCT domain, as we have described. If the first is positive and the second is negative, the pattern is +1 +1, and so on.

These problems can be solved by normalizing the extracted mark and reference mark to unit magnitude before computing the inner product between them. That is,

$$\tilde{\mathbf{v}} = \frac{\mathbf{v}}{|\mathbf{v}|}$$

$$\tilde{\mathbf{w}}_\mathbf{r} = \frac{\mathbf{w}_\mathbf{r}}{|\mathbf{w}_\mathbf{r}|}$$

$$z_{\text{nc}}(\mathbf{v}, \mathbf{w}_\mathbf{r}) = \sum_i \tilde{\mathbf{v}}[i]\tilde{\mathbf{w}}_\mathbf{r}[i]. \tag{3.23}$$

We refer to this measure as, simply, *normalized correlation*.

Geometric Interpretation

The detection region obtained by applying a threshold to normalized correlation is very different from that obtained by applying a threshold to linear correlation. Whereas linear correlation leads to a detection region containing all points on one side of a hyperplane, normalized correlation leads to a *conical* detection region. This follows from the fact that the inner product of two vectors is equal to the product of their Euclidian lengths and the cosine of the angle between them, rendering

$$\mathbf{v} \cdot \mathbf{w}_\mathbf{r} = |\mathbf{v}||\mathbf{w}_\mathbf{r}| \cos(\theta), \tag{3.24}$$

where θ is the angle between $\mathbf{v}$ and $\mathbf{w}_\mathbf{r}$. Thus, the normalized correlation between two vectors is just the cosine of the angle between them. Applying a threshold to this value is equivalent to applying a threshold to the angle between the vectors, rendering

$$\frac{\mathbf{v} \cdot \mathbf{w}_\mathbf{r}}{|\mathbf{v}||\mathbf{w}_\mathbf{r}|} > \tau_{\text{nc}} \quad \Leftrightarrow \quad \theta < \tau_\theta, \tag{3.25}$$

where

$$\tau_\theta = \cos^{-1}(\tau_{\text{nc}}). \tag{3.26}$$

Thus, the detection region for a given reference vector, $\mathbf{w}_\mathbf{r}$, is an N-dimensional cone (N-cone) centered on $\mathbf{w}_\mathbf{r}$ and subtending an angle of $2\tau_\theta$.

Figure 3.21 illustrates the detection region for one reference vector, $\mathbf{w}_\mathbf{r}$, and threshold, τ_{nc}. The figure shows a slice of marking space. The x-axis of the figure is aligned with the reference mark. The y-axis is an arbitrary direction orthogonal to the reference vector. The shaded region shows the set of points on the plane that will be detected as containing the reference mark, $\mathbf{w}_\mathbf{r}$. This region, being a two-dimensional slice of an N-dimensional cone, is the same for all planes that contain $\mathbf{w}_\mathbf{r}$. In other words, the figure would not change if we chose a different direction for the y-axis (provided it was still orthogonal to the

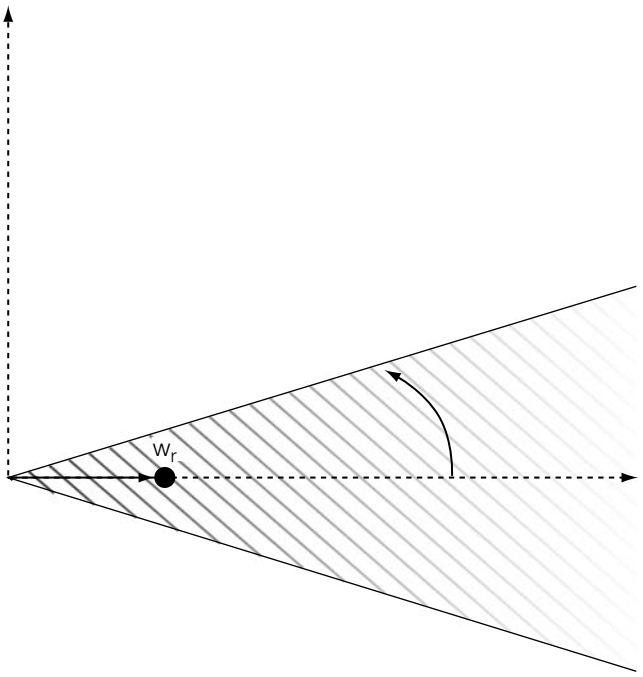


FIGURE 3.21

Detection region obtained by thresholding normalized correlation, where τ_θ is given by Equation 3.26.

x-axis). Note that the threshold illustrated here is very high. Lower thresholds lead to wider cones.

It may seem surprising that the detection region for normalized correlation is such a completely different shape from that for linear correlation. After all, if the vectors have zero means, we might think that the sample standard deviation, $s_{\mathbf{v}}$, of the extracted mark, $\mathbf{v}$, should be a reasonable approximation of the *statistical* standard deviation, $\sigma_{\mathbf{v}}$, of the random process that generates elements of extracted vectors. Recall that the statistical standard deviation is the square root of the expected value of $(\mathbf{v} - \mu_{\mathbf{v}})^2$, where $\mu_{\mathbf{v}}$ is the expected value of $\mathbf{v}$. The sample standard deviation, $s_{\mathbf{v}} = |\widetilde{\mathbf{v}}|/\sqrt{N}$, approaches the statistical standard deviation as N approaches infinity.

If we knew the statistical standard deviation, and we normalized linear correlation by it instead of by the $|\widetilde{\mathbf{v}}|$ estimate, we would obtain a planar detection region, because we would be just scaling linear correlation by a constant factor. The difference between the planar detection region for scaled linear correlation and the conical detection region for normalized correlation illustrates the type of odd phenomenon that can arise when data is normalized by statistical parameters estimated from the data itself.

Equivalent Detection Methods

It is sometimes suggested that linear correlation can be used as a detection measure, but that the detector's threshold should be scaled by the magnitude of the extracted mark. This is equivalent to the use of a normalized correlation detection measure. To see this, we begin by noting that the same behavior can be obtained by dividing the correlation by the extracted vector's magnitude, which leads to a detection measure of [96]:

$$z_1(\mathbf{v}, \mathbf{w_r}) = \frac{\mathbf{v} \cdot \mathbf{w_r}}{|\mathbf{v}|}. \tag{3.27}$$

The only difference between this and normalized correlation, as defined previously, is that here we have not normalized the magnitude of the reference mark. However, reference marks are usually constrained to have some constant magnitude, in which case this z_1 measure differs from z_{nc} by at most a constant factor.

3.5.3 Correlation Coefficient

The final form of correlation we make use of in this book is the *correlation coefficient,* obtained by subtracting out the means of two vectors before computing the normalized correlation between them, as follows:

$$\begin{aligned} \tilde{\mathbf{v}} &= \mathbf{v} - \overline{\mathbf{v}} \\ \tilde{\mathbf{w}}_\mathbf{r} &= \mathbf{w_r} - \overline{\mathbf{w}}_\mathbf{r} \\ z_{cc}(\mathbf{v}, \mathbf{w_r}) &= z_{nc}(\tilde{\mathbf{v}}, \tilde{\mathbf{w}}_\mathbf{r}). \end{aligned} \tag{3.28}$$

This provides robustness against changes in the DC term of a Work, such as the addition of a constant intensity to all pixels of an image.

Geometric Interpretation

Normalized correlation and the correlation coefficient bear a simple, geometric relationship to each other: The correlation coefficient between two vectors in N-space is just the normalized correlation between those two vectors after projection into an $(N-1)$-space. That is, the subtraction of the two vectors' means amounts to a projection into a lower-dimensional space. This can be seen by viewing it as vector subtraction. For example, if $\mathbf{v}$ is a three-dimensional vector, $[x, y, z]$, then $\mathbf{v} - \overline{\mathbf{v}} = [x, y, z] - [\overline{\mathbf{v}}, \overline{\mathbf{v}}, \overline{\mathbf{v}}]$. The vector $[\overline{\mathbf{v}}, \overline{\mathbf{v}}, \overline{\mathbf{v}}]$ describes the point on the diagonal of the coordinate system that lies closest to $\mathbf{v}$. Thus, the result of this subtraction is a vector that is orthogonal to the diagonal. This means that the resulting vector lies in an $(N-1)$-space that is orthogonal to the diagonal of the N-dimensional coordinate system. A simple, two-dimensional example is shown in Figure 3.22.

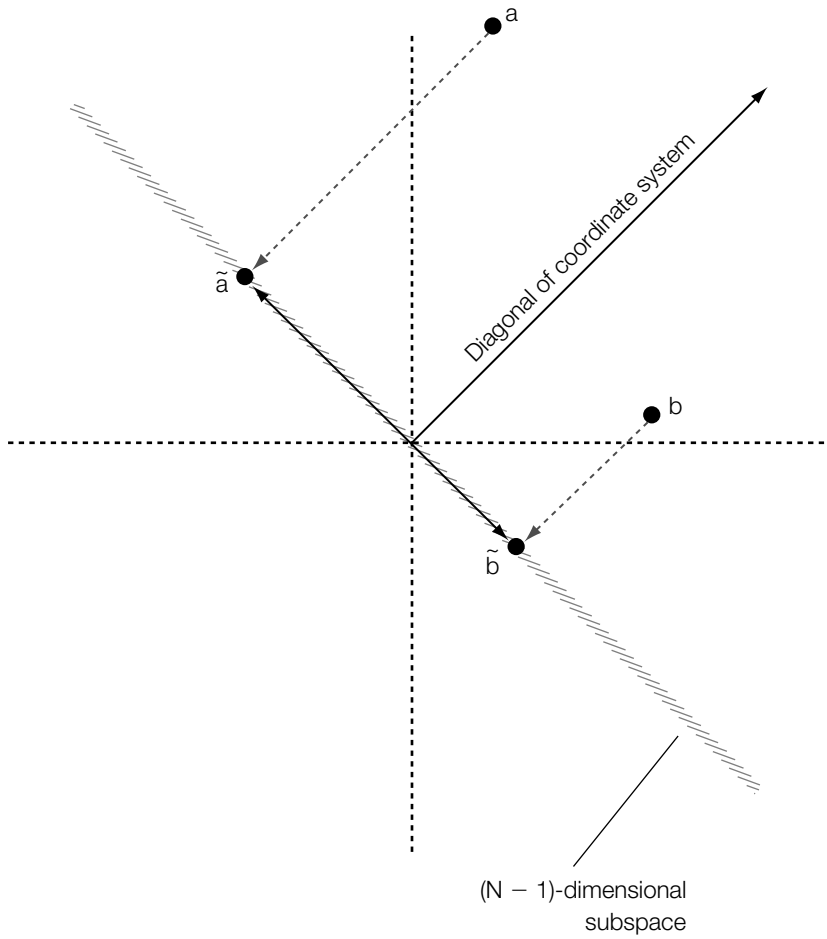


FIGURE 3.22

Subtraction of mean *x*, *y* values from two two-dimensional vectors. This has the effect of projecting along the diagonal of the coordinate system into a one-dimensional subspace.

Because the correlation coefficient is just normalized correlation after a projection, we often use the two measures interchangeably, deriving principles based on normalized correlation and illustrating them with systems that employ the correlation coefficient.

Equivalent Detection Methods

If we scale linear correlation by the sample standard deviation of an extracted vector, we obtain the following detection measure:

$$z_2(\mathbf{v}, \mathbf{w_r}) = \frac{\mathbf{v} \cdot \mathbf{w_r}}{s_{\mathbf{v}}} \tag{3.29}$$

Although this is only slightly different from the practice of scaling by the extracted mark's magnitude (detection measure z_1), it leads to detection by correlation coefficient rather than normalized correlation.

The sample standard deviation, $s_{\mathbf{v}}$, of the extracted vector, $\mathbf{v}$, can be expressed as

$$\begin{aligned} s_{\mathbf{v}} &= \sqrt{\frac{1}{N}\sum_{i}^{N}(v[i]-\overline{v}[i])^2} \\ &= \frac{1}{\sqrt{N}}|\mathbf{v}-\overline{\mathbf{v}}| \\ &= \frac{1}{\sqrt{N}}|\widetilde{\mathbf{v}}|, \end{aligned} \tag{3.30}$$

where $\widetilde{\mathbf{v}}$ is as defined in Equation 3.28. Thus, the detection metric in such a system is given by

$$z_2(\mathbf{v}, \mathbf{w_r}) = \sqrt{N}\frac{\mathbf{v}\cdot\mathbf{w_r}}{|\widetilde{\mathbf{v}}|}. \tag{3.31}$$

If the reference marks are constrained to have constant magnitude and zero mean (which is common), this is equivalent to the correlation coefficient, in that $\mathbf{w_r} = \widetilde{\mathbf{w}}_\mathbf{r}$ and $\mathbf{v}\cdot\widetilde{\mathbf{w}}_\mathbf{r} = \widetilde{\mathbf{v}}\cdot\widetilde{\mathbf{w}}_\mathbf{r}$ (because the mean of $\mathbf{v}$ has no effect on its dot product with a zero-mean vector).

3.6 SUMMARY

This chapter has laid the groundwork for the technical discussion of the succeeding chapters by describing various ways of modeling watermarking systems. The main topics covered include the following.

- We described three models of watermarking systems based on the traditional model of a communications channel. These differ in how the cover Work is incorporated into the system.
  - In the basic model, the cover Work is considered noise added during transmission of the watermark signal.
  - In models of watermarking as communications with side information at the transmitter, the cover Work is still considered noise, but the watermark encoding process is provided with its value as side information.
  - In models of watermarking as multiplexing, the cover Work and the watermark are considered two messages multiplexed together for reception by two different "receivers": a human and a watermark detector, respectively.

- Works can be viewed as points in a high-dimensional *media space*. Within this space there are the following probability distributions and regions of interest:
  - The *distribution of unwatermarked Works* indicates how likely each Work is.
  - The *region of acceptable fidelity* is a region in which all Works appear essentially identical to a given cover Work.
  - The *detection region* describes the behavior of the detection algorithm.
  - The *embedding distribution* or *embedding region* describes the effects of an embedding algorithm.
- The *distortion distribution* indicates how Works are likely to be distorted during normal usage.
- Part of a watermarking system can be viewed as an *extraction process* that projects or distorts media space into a *marking space*. The rest can then be viewed as a simpler watermarking system that operates in marking space rather than in media space.
- Many watermarking systems fall into the class of *correlation-based systems,* in which the detector uses some form of correlation as a detection metric. This is true even of many systems not explicitly described as using correlation-based detection.
- The following are the main forms of correlation discussed in this book:
  - Linear correlation, computed as the inner product between two vectors divided by their dimensionality. Applying a threshold to this measure leads to a detection region bounded by a hyperplane.
  - Normalized correlation, computed by normalizing two vectors to unit magnitude before taking their inner product. Applying a threshold to this measure leads to a conical detection region.
  - Correlation coefficient, computed by subtracting the means from two vectors before computing their normalized correlation. This is equivalent to computing normalized correlation in a space with one fewer dimension. We often use the terms *normalized correlation* and *correlation coefficient* interchangeably.

This page intentionally left blank

CHAPTER 4

Basic Message Coding

The example systems described in the previous chapter embed only one bit of information. In practice, most applications require more information than this. Thus, we now consider a variety of methods for representing messages with watermarks. In this chapter, we concentrate on methods that result in simple extensions of the watermarking systems described so far and that are mostly straightforward applications of methods used in traditional communications. Chapter 5 focuses on optimal use of side information (see Section 3.3.2), which entails a less conventional coding method.

Although we do not intend this chapter to serve as a comprehensive introduction to communications theory, we have written it assuming the reader has no prior knowledge of this field. Those who are familiar with the field may wish to skim.

We begin, in Section 4.1, by examining the problem of mapping messages into *message marks* (i.e., vectors in marking space). The simplest way to do this, which we refer to as *direct message coding,* is to predefine a separate reference mark for each message. However, this approach is not feasible for more than a small data payload. Thus, we discuss the idea of first representing messages with sequences of symbols, and subsequently mapping the symbol sequences into vectors in marking space. The section ends with an example of a simple watermarking system that uses blind embedding to embed 8-bit messages.

The performance of the system that ends Section 4.1 is not as good as we might want, and this leads us, in Section 4.2, to explore the field of error correction coding. We present an example trellis code and show that it significantly improves the performance of our watermarking systems.

Finally, in Section 4.3, we turn to the problem of determining whether or not a message is present in a Work. When messages are represented with sequences of symbols, there are a number of ways to do this. We discuss three approaches and give an example system that uses one of them.

4.1 MAPPING MESSAGES INTO MESSAGE VECTORS

The problem of multimessage watermarking can be cast as a problem of mapping messages into watermark vectors, and vice versa. In traditional communications systems, the analogous mapping—between messages and transmitted signals—is often performed on two levels: *source coding* maps messages into sequences of symbols and *modulation* maps sequences of symbols into physical signals. Watermarking systems can be constructed in the same way, with modulation taking the form of a mapping between sequences of symbols and watermark vectors.

We begin this discussion with the simplest approach, which assigns a predefined, unique watermark vector to each message. This approach can be viewed either as involving no intermediate representation of messages with symbols or as assigning a single symbol to each message. In our discussion, we take the former view and, accordingly, refer to the approach as *direct message coding*. In discussing direct message coding, we introduce the concept of *code separation* and examine desirable properties of codes for correlation-based watermarking systems.

We then move on to the subject of symbol sequences. Our discussion of this topic focuses on methods of mapping symbol sequences into watermark vectors (i.e., modulation). Four common methods are presented: *time-division multiplexing* represents symbols with temporally disjoint watermark patterns; *space-division multiplexing* represents symbols with spatially disjoint patterns; *frequency-division multiplexing* represents symbols with patterns that are disjoint in the frequency domain; and *code-division multiplexing* represents symbols with patterns that overlap in time/space and frequency but have zero correlation with one another. We argue that all of these can be considered special cases of code-division multiplexing.

4.1.1 Direct Message Coding

The most straightforward approach to message coding is the assignment of a unique, predefined message mark to represent each message. Let the set of messages be denoted $\mathcal{M}$ and the number of messages in the set $|\mathcal{M}|$. We design a set of $|\mathcal{M}|$ message marks, $\mathcal{W}$, each of which is associated with a message. We use $\mathcal{W}[m]$ to denote the message mark associated with message m, $m \in \mathcal{M}$. To watermark a Work with message m, the embedder simply embeds message mark $\mathcal{W}[m]$. We refer to this as the *direct* coding method.

At the detector, detection values are computed for each of the $|\mathcal{M}|$ message marks. The most likely message is the one that corresponds to the message mark with the highest detection value. If this value is less than the threshold, the detector reports that no watermark is present. If the value is greater than

the threshold, the detector outputs the index as the detected message. This is known as *maximum likelihood detection*.

How should we design the message marks for our messages? Obviously, each message mark should be chosen to have good behavior with respect to predictable false positive rate, fidelity, robustness, and so on. However, in addition to these issues, we must also consider the likelihood that one message mark will be confused with another. Before being detected, the message mark is likely to be corrupted, either by the embedding algorithm (due to clipping, round-off, perceptual shaping, and so on) or by subsequent processing. If this corruption changes the message mark too much, the watermark might be erroneously decoded as a different message. Thus, we wish to choose a set of message marks that have good *code separation* (i.e., marks that are far apart from one another in marking space).

If we are using either linear correlation or normalized correlation for our detection metric, achieving good code separation means ensuring that the message marks have low correlations with one another. The best case, in fact, is to arrange for each message mark to have a *negative* correlation with each of the others, because this would mean that embedding one message mark would *decrease* the correlations between the watermarked Work and the other message marks, thus decreasing the chances that a different message mark will be detected.

For a system with only two messages, such as the example systems described so far, the ideal is achieved by defining one message mark to be the negative of the other. Thus, the two marks have a correlation of –1 and achieve maximal separation. For maximal separation in a system with three messages, the angle between the marks should be 120 degrees, as shown in Figure 4.1. This figure shows three message mark vectors in a two-dimensional plane of marking space, along with their respective linear correlation detection regions for an arbitrary threshold. In the general case, the problem of designing an optimal set of $|\mathcal{M}|$ N-dimensional message marks is equivalent to the problem of placing $|\mathcal{M}|$ points on the surface of an N-dimensional sphere (N-sphere) such that the distance between the two closest points is maximized.

If the number of messages is large compared to the dimensionality of marking space, it is well established that randomly generated codes usually result in good code separation [89]. This is illustrated in Figure 4.2, which shows the result of a simple experiment in which 10,000 three-message codes were randomly generated in a three-dimensional space. In each code, the message marks were chosen from an i.i.d. Gaussian distribution. The figure is a histogram of the average angle between all pairs of message vectors in each code. Note that the most common average angles were close to the optimal angle of 120 degrees. This effect becomes more pronounced in higher dimensions.

On the other hand, if the number of messages is small compared to the dimensionality of marking space, randomly generated message vectors are

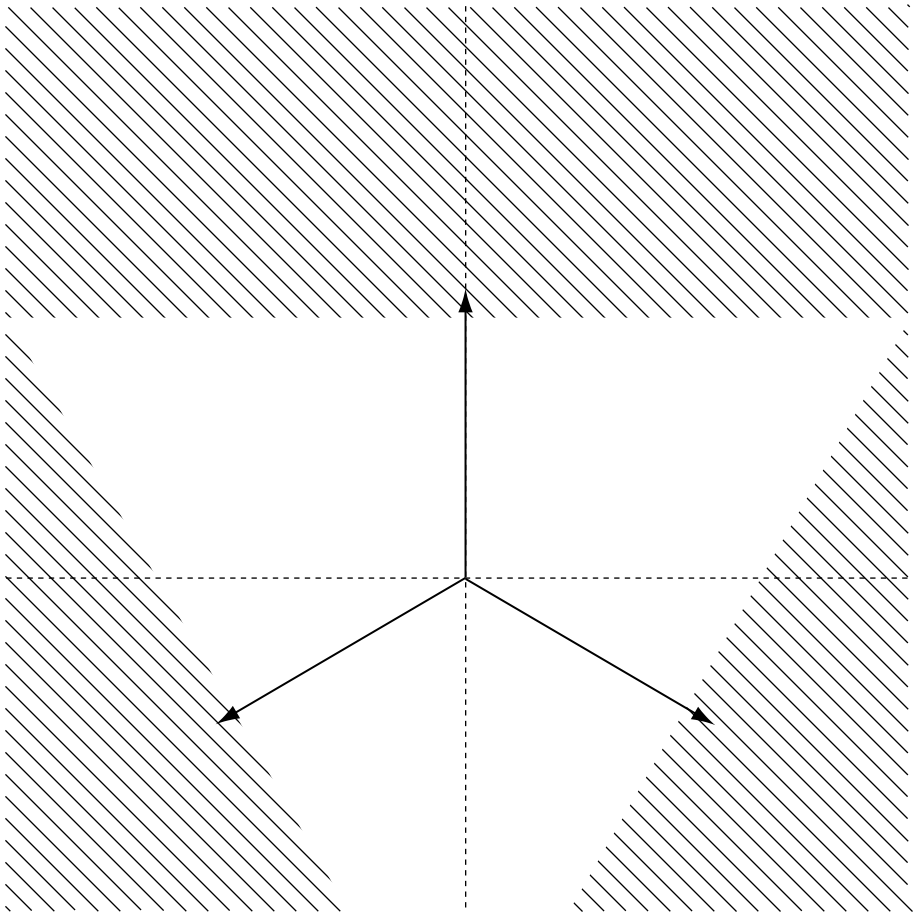


FIGURE 4.1

Optimal arrangement of detection regions for three messages using linear correlation detection.

likely to be orthogonal to one another. This is illustrated in Figure 4.3, which shows the results of an experiment similar to that in Figure 4.2, except that the three-message codes were generated in a 256-dimensional space. In this case, the most common average angle was 90 degrees.

An interesting and useful property of orthogonal message marks is that in a linear correlation–based watermarking system more than one message can be embedded in a single Work. Because each message mark has no effect on correlation with other message marks, embedding an additional mark has no effect on the detectability of any mark previously embedded. This situation is illustrated

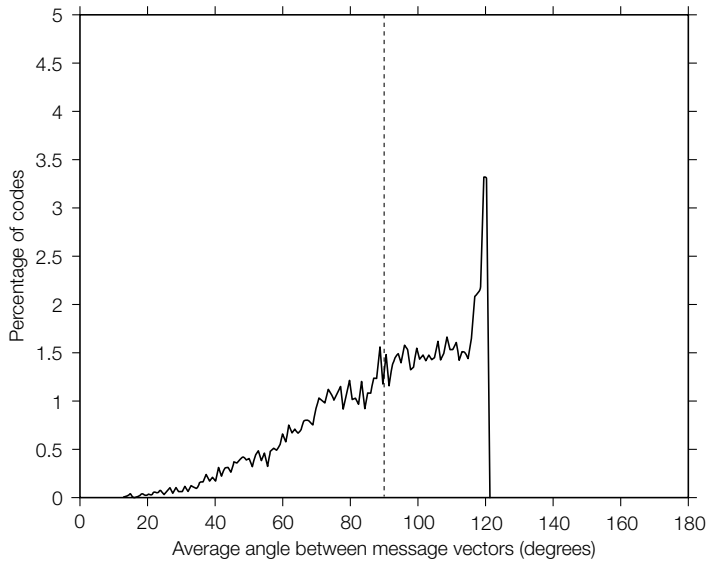


FIGURE 4.2

Distribution of average angles between three-message vectors in randomly generated three-dimensional codes.

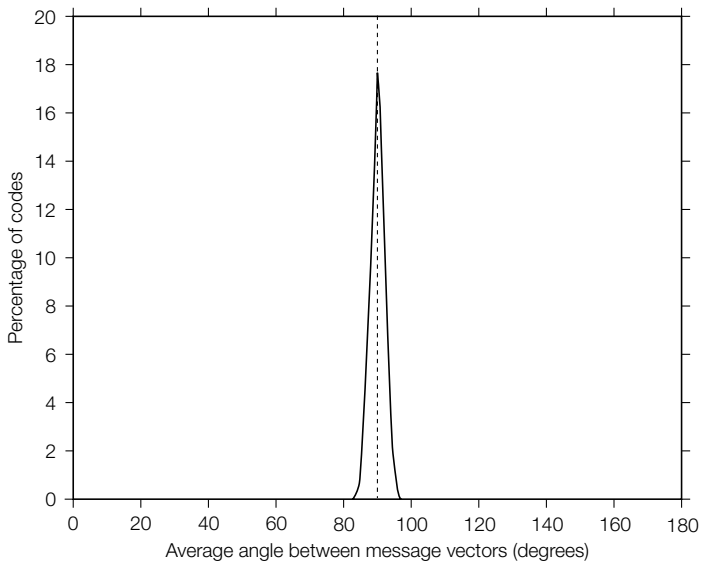


FIGURE 4.3

Distribution of average angles between three-message vectors in randomly generated 256-dimensional codes.

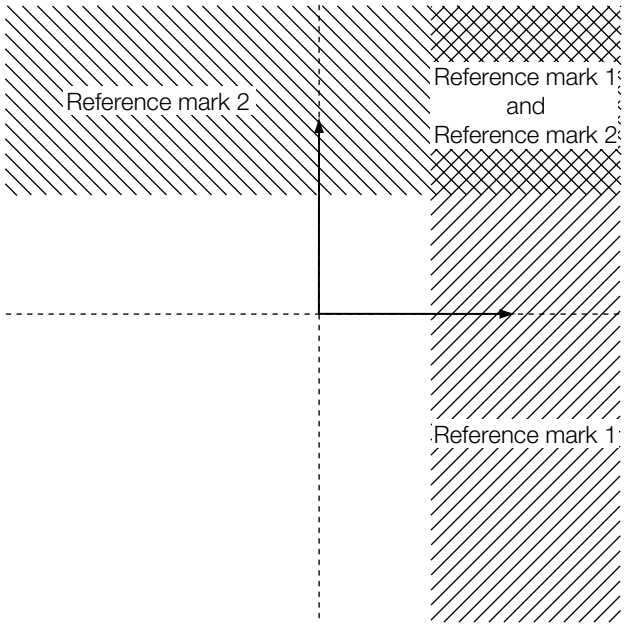


FIGURE 4.4

Detection regions for two orthogonal watermarks, showing overlap where both watermarks are detected.

in Figure 4.4, which shows two orthogonal (i.e., zero correlation) message vectors in a two-dimensional plane of marking space, together with their respective detection regions. The plane is actually divided into *three* detection regions: one for message 1, one for message 2, and one for both messages. In the following section, we take advantage of this property to encode messages with combinations of marks.

4.1.2 Multisymbol Message Coding

Direct message coding is effective, but does not scale well. Because the detector must compute the detection value for each of $|\mathcal{M}|$ reference marks, the approach is not feasible when $|\mathcal{M}|$ is large. For example, if we wish to embed just 16 bits of information, the detector would have to store and compare against 65,536 different reference marks, taking up to 65,536 times the computation required for a 0-bit watermark. Encoding on the order of 100 bits in this manner is completely impractical.

This problem can be dramatically reduced by first representing each message with a sequence of symbols, drawn from an *alphabet,* $\mathcal{A}$. Each symbol in a

sequence can then be embedded and detected independently with its own reference mark.

If each message is represented with a sequence of L separate symbols, drawn from an alphabet of size $|\mathcal{A}|$, we can represent up to $|\mathcal{A}|^L$ different messages. Detecting these messages requires only a fraction of the processing required with direct message coding. For example, suppose we use an alphabet of $|\mathcal{A}| = 4$ symbols and a message length of $L = 8$. This can represent up to $|\mathcal{A}|^L = 65{,}536$ different messages, or 16 bits of information. At the detector, detection of each of the eight symbols requires comparison with each of the four reference marks, for a total of 32 comparisons, which is considerably fewer than the 65,536 required in a system that represents each message with its own reference mark.

The simplest method of embedding a sequence of symbols is to independently embed a reference mark for each symbol. In the type of systems discussed so far, this would amount to adding L reference marks to the cover Work. Alternatively, and more generally, we can think of the embedder as first modulating the sequence of symbols into a single message mark, and then embedding this message mark. In the following, we discuss a few such forms of modulation.

Time- and Space-Division Multiplexing

One straightforward method of modulating symbol sequences is to divide the Work into disjoint regions, either in space or time, and embed a reference mark for one symbol into each part. That is, the message mark is constructed by concatenating several reference marks.

For example, to embed a sequence of eight symbols into an audio clip of length l samples, we would use reference marks of length $l/8$. To embed four symbols into an image of dimensions w pixels × h pixels, we would use reference marks of dimensions $w/2 \times h/2$.

These methods are referred to as *time-division* and *space-division multiplexing,* respectively. Note that time-division multiplexing is essentially the same as the standard method of communicating over temporal channels.

Frequency-Division Multiplexing

Alternatively, we can divide the Work into disjoint bands in the frequency domain and embed a reference mark for one symbol in each. That is, the message mark is constructed by adding together several reference marks of different frequencies. This is referred to as *frequency-division multiplexing*.

One way to implement frequency-division multiplexing is to define a watermark extraction function that involves a frequency transform, and then design an embedder and detector according to the structures described in Section 3.4.2. The simple watermark embedder employed in this structure should divide its input into segments and embed one symbol in each. Thus,

the simple embedder can be essentially identical to one designed for a simple time- or space-division multiplexing system.

Code-Division Multiplexing

As an alternative modulation method, we can take advantage of the fact that several uncorrelated reference marks embedded in the same Work have no effect on one another in a linear correlation system. This leads to a system analogous to *code-division multiplexing* in spread spectrum communications.

If we are representing messages with sequences of L symbols drawn from an alphabet of size $|\mathcal{A}|$, we define a set of $L \times |\mathcal{A}|$ reference marks, $\mathcal{W}_{AL}$. Each mark corresponds to a given symbol at a given index in the sequence. Let $\mathcal{W}_{AL}[i, s]$ correspond to symbol s at location i. The message mark is constructed by adding together the appropriate reference marks.

For example, Figure 4.5 illustrates a possible set of 8 × 8 reference marks for use in an image watermarking system such as E_BLK_BLIND/D_BLK_CC, which uses a marking space arrived at by averaging 8 × 8 image blocks. The table of 8 × 8 marks, $\mathcal{W}_{AL}[1 \ldots 5, 1 \ldots 4]$, can be used to modulate sequences of $L = 5$ symbols drawn from an alphabet of size $|\mathcal{A}| = 4$. The figure shows the encoding of the symbol sequence 3, 1, 4, 4, 2; namely,

$$\mathbf{w}_m = \mathcal{W}_{AL}[1, 3] + \mathcal{W}_{AL}[2, 1] + \mathcal{W}_{AL}[3, 4] + \mathcal{W}_{AL}[4, 4] + \mathcal{W}_{AL}[5, 2], \tag{4.1}$$

where $\mathbf{w}_m$ is the resulting message mark.

We need to ensure that all reference marks added into $\mathbf{w}_m$ are close to orthogonal. Each mark represents a symbol in a distinct symbol position, so that two marks will be added together only if they correspond to symbols in different positions. For example, we might add $\mathcal{W}_{AL}[1, 3]$ and $\mathcal{W}_{AL}[2, 1]$ in a single message mark, but we would never embed both $\mathcal{W}_{AL}[1, 3]$ and $\mathcal{W}_{AL}[1, 1]$. Thus, we can ensure that the embedded marks are orthogonal or nearly orthogonal by ensuring only that for any two symbols, a and b (including the case where $a = b$), $\mathcal{W}_{AL}[i, a] \cdot \mathcal{W}_{AL}[j, b] \approx 0$ if $i \neq j$.

We also need to ensure that we do not confuse one symbol for another in a given location. This means that if $a \neq b$, the two reference marks used to encode them at index i, $\mathcal{W}_{AL}[i, a]$ and $\mathcal{W}_{AL}[i, b]$, must be maximally distinguishable. As discussed earlier, this objective is best served by making these reference marks have a negative correlation.

In some cases, we may be concerned with the problem of registering a Work to counter temporal delay or geometric translation (see Chapter 9). Under these circumstances, we might need the reference marks for different symbol positions to be not only uncorrelated with one another but uncorrelated with *shifted* versions of one another. That is, we wish the marks for different symbol positions to have low *cross-correlations*. Otherwise, a temporal or geometric shift might cause the reference mark for a symbol in one position to be confused with that for a different symbol and/or position. For the one-dimensional case of temporal shifting, this is a well-studied problem. Several

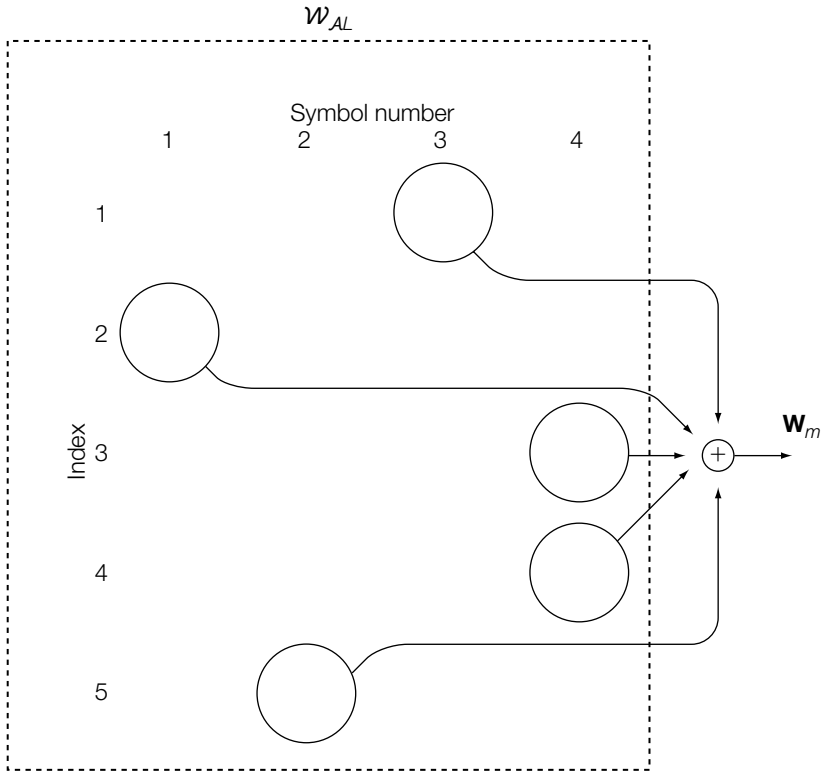


FIGURE 4.5

Example of code-division multiplexing using 8×8 reference marks. Each row of $\mathcal{W}_{AL}$ corresponds to one position in the symbol sequence. Each column corresponds to one symbol. Thus, there are five representations of each symbol; one for each possible position in the symbol sequence. Selected reference marks are added to encode a message.

sets of code vectors that exhibit low cross-correlations have been developed, including *gold-sequences* and *m-sequences*. Many of these are described in Sarwate and Pursley [358]. For geometric shifting, the problem is less well studied, but some efforts have been made to extend one-dimensional sequences to two-dimensional patterns [109, 280, 421–423].

Equivalence of Code-Division Multiplexing to Other Approaches

Time-, space-, and frequency-division multiplexing can be viewed as special cases of code-division multiplexing. Consider a length $L = 8$ message modulated for embedding in an audio clip using time-division multiplexing. This is done by concatenating eight short reference marks. We can, alternatively, view each of the added reference marks as being added to the *entire* message mark if we just pad the marks with zeros. This point is illustrated in Figure 4.6. At the

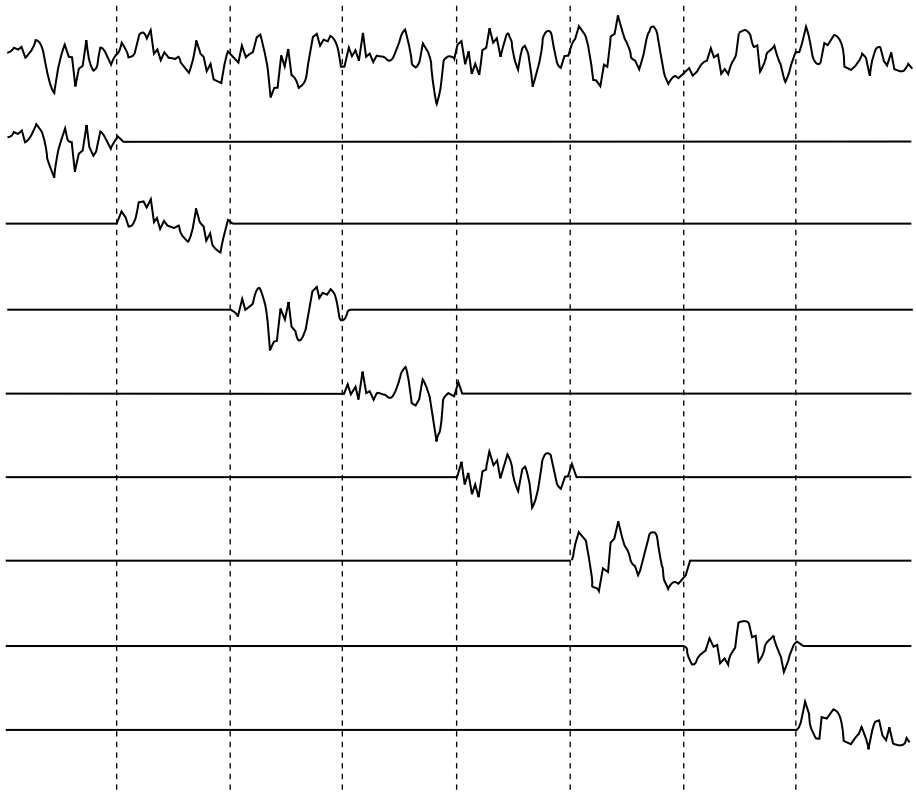


FIGURE 4.6

Interpretation of time-division multiplexing as code-division multiplexing.

top of the figure is a sequence of eight short reference marks that might be embedded in an audio signal to represent a sequence of eight symbols. Below this are eight full-length reference marks that add up to the signal at the top. Note that these full-length marks are orthogonal to one another, in that only one of them is nonzero in any given element.

Figure 4.7 is the corresponding illustration for space-division multiplexing. Thus, in a sequencing system, each reference mark has nonzero values in only $1/L$ of its elements, and $\mathcal{W}_{AL}[i, s]$ is just a shifted version of $\mathcal{W}_{AL}[j, s]$ if $i \neq j$.

Similarly, frequency-division multiplexing can be implemented by defining patterns that are band limited in the frequency domain, and then converting them to the temporal or spatial domains. If the frequency transform used is linear (such as the Fourier or discrete cosine transforms), then patterns that do not overlap in the frequency domain will have zero correlation in the temporal or spatial domains, even though they overlap in time or space.

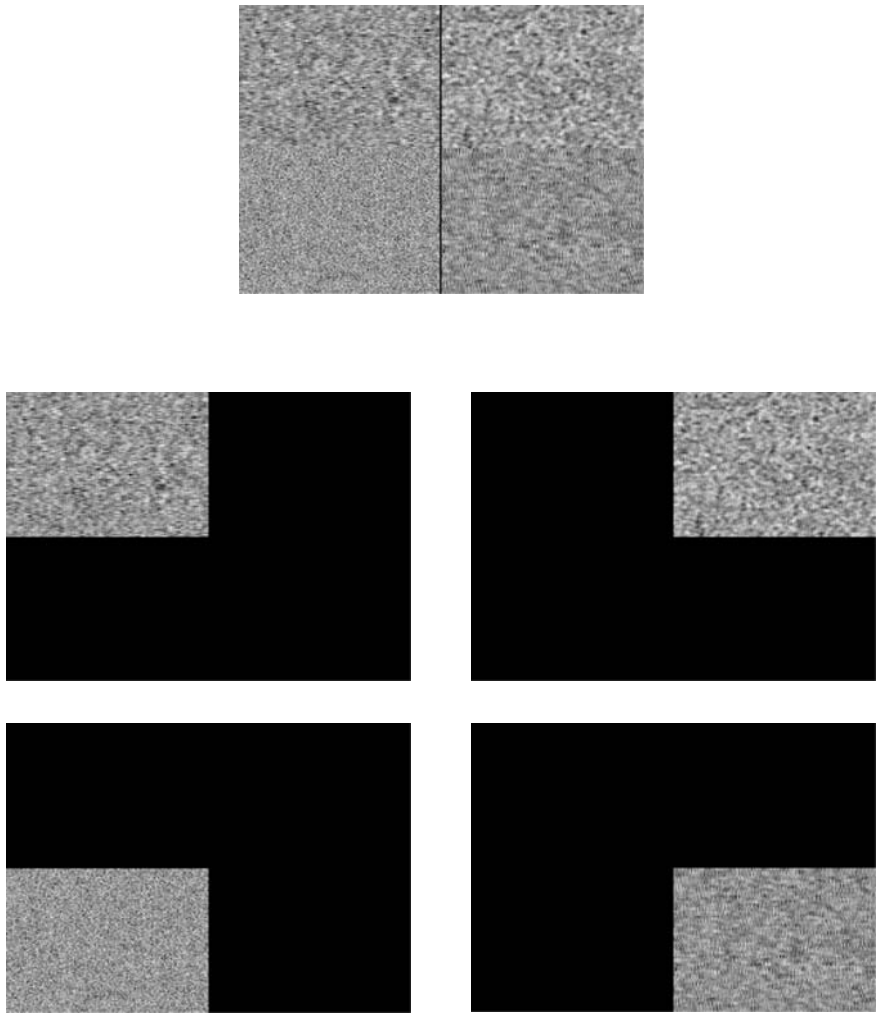


FIGURE 4.7

Interpretation of space-division multiplexing as code-division multiplexing.

INVESTIGATION

Simple 8-Bit Watermarks

We now present a simple example of a watermarking system using code division to embed integer messages, represented as sequences of 8 bits. That is, the message length, L, is 8, and the alphabet size, $|\mathcal{A}|$, is 2. This system is based on the E_BLIND/D_LC system. We refer to this new system as E_SIMPLE_8/D_SIMPLE_8.

System 4: E_SIMPLE_8/D_SIMPLE_8

In the E_SIMPLE_8/D_SIMPLE_8 system, the mapping of a message into a sequence of symbols is simply a mapping of the message into a bit string. To modulate a message, the i^{th} bit of the string is then mapped into $\mathcal{W}_{AL}[i, 1]$ if it is 1, or $\mathcal{W}_{AL}[i, 0]$ if it is 0.

The two reference patterns for a given location must be maximally distinguishable from each other. As pointed out previously, two vectors are maximally distinguishable when one is the negative of the other. Therefore, we can construct an optimal binary system by designing a single reference pattern, $\mathbf{w}_{\mathbf{r}i}$, for each bit location, i, and using that pattern to encode a 1, or the negation of that pattern to encode a 0. In other words, for any bit location, i, $\mathcal{W}_{AL}[i, 1] = \mathbf{w}_{\mathbf{r}i}$ and $\mathcal{W}_{AL}[i, 0] = -\mathbf{w}_{\mathbf{r}i}$.

Each of the base reference patterns, $\mathbf{w}_{\mathbf{r}1}, \mathbf{w}_{\mathbf{r}2}, \ldots, \mathbf{w}_{\mathbf{r}8}$, is generated pseudo-randomly according to a given seed, which can be considered a watermark key. They are drawn from i.i.d. Gaussian distributions. We normalize each pattern to have zero mean. To ensure that our test results are comparable with those of other systems in this book, we normalize the modulated message pattern, $\mathbf{w}_m$, to have unit variance (it already has zero mean because its component reference patterns have zero means).

Thus, the message pattern, $\mathbf{w}_m$, that encodes a given message, m, expressed as a sequence of bits, $m[1]$, $m[2] \ldots m[s]$, is given by

$$\mathbf{w}_{mi} = \begin{cases} \mathbf{w}_{\mathbf{r}i} & \text{if } m[i] = 1 \\ -\mathbf{w}_{\mathbf{r}i} & \text{if } m[i] = 0 \end{cases} \tag{4.2}$$

$$\mathbf{w}_{\text{tmp}} = \sum_i \mathbf{w}_{mi} \tag{4.3}$$

$$\mathbf{w}_m = \frac{\mathbf{w}_{\text{tmp}}}{s_{\mathbf{w}_{\text{tmp}}}}, \tag{4.4}$$

where $\mathbf{w}_{\text{tmp}}$ is just the version of the message mark before normalization, and $s_{\mathbf{w}_{\text{tmp}}}$ is its sample standard deviation.

The E_SIMPLE_8 embedder computes the message mark according to Equation 4.4 and embeds it using blind embedding. Thus, the watermarked image, $\mathbf{c}_{\mathbf{w}}$, is given by

$$\mathbf{c}_{\mathbf{w}} = \mathbf{c}_{\mathbf{o}} + \alpha \mathbf{c}_m, \tag{4.5}$$

where $\mathbf{c}_{\mathbf{o}}$ is the original image, and α is a strength parameter input by the user.

The D_SIMPLE_8 detector correlates the received image, $\mathbf{c}$, against each of the eight base reference patterns, $\mathbf{w}_{\mathbf{r}1}, \ldots, \mathbf{w}_{\mathbf{r}8}$, and uses the sign of each correlation to determine the most likely value for the corresponding bit. This yields the decoded message. The detector does not distinguish between marked and unwatermarked images. If an image is not watermarked, the output message is essentially random. (Section 4.3 discusses methods of detecting the presence of multisymbol watermarks.)

Experiments

To test the E_SIMPLE_8/D_SIMPLE_8 system, we embedded six different messages in each of 2,000 images. The messages embedded were m = 0, 255, 101, 154, 128, and 127. This resulted in 12,000 watermarked images.

The embedding strength, α, had to be higher than 1, which we used in most of our preceding experiments, as we are effectively embedding eight different watermarks. Because the message pattern is scaled to have unit standard deviation, each of the eight patterns embedded has an effective strength of roughly $\alpha/\sqrt{8}$. If $\alpha = 1$, these individual patterns are not reliably embedded and detected. Thus, we used an embedding strength of $\alpha = 2$.

Out of the 12,000 messages embedded, a total of 26 messages were not correctly detected. Although this may sound like a small number, it probably would not be acceptable for many applications. The problem lies in the amount of code separation achieved using the simple coding scheme of this system. That is, the maximum correlation between two different message vectors is high enough that when the image is added as "noise," one message vector gets confused with another. To reduce this problem, we can turn to the field of error correction coding.

4.2 ERROR CORRECTION CODING

If every possible symbol sequence of a given length is used to represent a message, it is easily demonstrated that, using code-division modulation, some of the resulting vectors will have poor code separation. This problem motivates the introduction of error correction codes. We describe trellis codes and Viterbi decoding as an example.

4.2.1 The Problem with Simple Multisymbol Messages

In direct message coding, we can choose any set of message marks we wish. Thus, we can ensure that the angle between any pair of message marks is maximal. In any multisymbol system, however, code separation depends on the methods used for source coding and modulation. In the case of the type of system used by E_SIMPLE_8/D_SIMPLE_8, where every possible sequence of symbols represents a distinct message and sequences are modulated with some form of code division, there is a lower limit on the inner product, and, hence, an upper limit on the angle between the resulting message marks.

To see why this is the case, consider a system with an alphabet size, $|\mathcal{A}|$, of four and a message length, L, of three. Let

$$\mathbf{w}_{312} = \mathcal{W}_{AL}[1, 3] + \mathcal{W}_{AL}[2, 1] + \mathcal{W}_{AL}[3, 2] \tag{4.6}$$

be the message mark embedded in a Work to represent the symbol sequence 3, 1, 2. Now examine the inner product between this message mark and the

message mark that encodes the sequence 3, 1, 4 (which differs only in the last symbol); namely,

$$\mathbf{w}_{314} = \mathcal{W}_{AL}[1, 3] + \mathcal{W}_{AL}[2, 1] + \mathcal{W}_{AL}[3, 4]. \tag{4.7}$$

The inner product between these two is given by

$$\begin{aligned}\mathbf{w}_{312} \cdot \mathbf{w}_{314} &= \left(\mathcal{W}_{AL}[1, 3] + \mathcal{W}_{AL}[2, 1] + \mathcal{W}_{AL}[3, 2]\right) \\ &\quad \cdot \left(\mathcal{W}_{AL}[1, 3] + \mathcal{W}_{AL}[2, 1] + \mathcal{W}_{AL}[3, 4]\right) \\ &= \mathcal{W}_{AL}[1, 3] \cdot \left(\mathcal{W}_{AL}[1, 3] + \mathcal{W}_{AL}[2, 1] + \mathcal{W}_{AL}[3, 4]\right) \\ &\quad + \mathcal{W}_{AL}[2, 1] \cdot \left(\mathcal{W}_{AL}[1, 3] + \mathcal{W}_{AL}[2, 1] + \mathcal{W}_{AL}[3, 4]\right) \\ &\quad + \mathcal{W}_{AL}[3, 2] \cdot \left(\mathcal{W}_{AL}[1, 3] + \mathcal{W}_{AL}[2, 1] + \mathcal{W}_{AL}[3, 4]\right).\end{aligned} \tag{4.8}$$

Because all marks for one location are orthogonal to all marks for any other location, this reduces to

$$\begin{aligned}\mathbf{w}_{312} \cdot \mathbf{w}_{314} &= \mathcal{W}_{AL}[1, 3] \cdot \mathcal{W}_{AL}[1, 3] + \mathcal{W}_{AL}[2, 1] \cdot \mathcal{W}_{AL}[2, 1] \\ &\quad + \mathcal{W}_{AL}[3, 2] \cdot \mathcal{W}_{AL}[3, 4].\end{aligned} \tag{4.9}$$

Suppose all the marks are normalized to have unit variance. This means $\mathcal{W}_{AL}[1, 3] \cdot \mathcal{W}_{AL}[1, 3]$ and $\mathcal{W}_{AL}[2, 1] \cdot \mathcal{W}_{AL}[2, 1]$ both equal N (the dimensionality of marking space), while $\mathcal{W}_{AL}[3, 2] \cdot \mathcal{W}_{AL}[3, 4]$ is bounded from below by $-N$. Thus, $\mathbf{w}_{312} \cdot \mathbf{w}_{314} \geq N$, and the inner product between the two closest encoded messages cannot possibly be lower than this.

In general, the smallest possible inner product between two message marks that differ in b symbols is $N(L - 2b)$. Thus, as L increases, the message marks of the closest pair become progressively more similar.

4.2.2 The Idea of Error Correction Codes

We can solve this problem by defining a source coding system in which *not* every possible sequence of symbols corresponds to a message. Sequences that correspond to messages are referred to as *code words*. The sequences that do not correspond to messages are interpreted as corrupted code words. By defining the mapping between messages and code words in an appropriate way, it is possible to build decoders that can identify the code word closest to a given corrupted sequence (i.e., decoders that correct errors). Such a system is an *error correction code* (ECC).

Error correction codes are typically implemented by increasing the lengths of symbol sequences. Thus, to use an ECC encoder, we begin by representing our message in the straightforward way described previously. For example, if we have 16 possible messages, we could represent the message with a sequence of 4 bits. The ECC encoder, then, takes this sequence as input, and outputs a

longer sequence, say, 7 bits. Of all the $2^7 = 128$ possible 7-bit words, only 16 would be code words. The set of code words can be defined in such a way that if we start with one code word, we have to flip at least 3 bits to obtain a code word that encodes a different message. When presented with a corrupted sequence, the decoder would look for the code word that differs from it in the fewest number of bits. If only 1 bit has been flipped between encoding and decoding, the system will still yield the correct message.

Note that if we have a 7-bit code ($L = 7$) encoding 4-bit messages and ensuring that the code words of every pair differ in at least 3 bits ($b = 3$), the maximum inner product between any two code words will be $N(L - 2b) = N(7 - 2 \times 3) = N$. This is better than the performance we would get without error correction, where messages are coded with only 4 bits ($L = 4$) and two messages can differ in as few as 1 bit ($b = 1$). In that case, the maximum inner product would be $N(L - 2b) = N(4 - 2) = 2N$.

A less typical implementation of an error correction code increases the size of the alphabet, rather than the length of the sequences. For example, our 4-bit messages might be encoded with sequences of symbols drawn from an alphabet of size 4, rather than a binary alphabet of size 2. In principle, this would be equivalent to a system that lengthens 4-bit binary sequences to 8 bits. The only difference is the way in which we would implement subsequent modulation.

There are many error correction codes available. One of the simplest, and earliest, is the *Hamming code* [352], which ensures that any two coded messages always differ in at least 3 bits, and allows correction of single-bit errors. This type of code can be used to realize the 7-bit coding of 4-bit messages described previously. More sophisticated codes, such as *BCH* [352] and *trellis codes* [427], allow a greater number of errors to be corrected. Some of the best-performing codes fall into the class of *turbo codes* [37], which a few researchers have begun using to encode watermark messages [124, 326].

These codes are often described in terms of methods of correcting symbol errors. Different codes are suitable for different types of errors. For example, *random* errors are well handled by Hamming codes, whereas *burst* errors (errors in groups of consecutive symbols) are better handled by BCH codes. However, because all code words are ultimately modulated, we can view all codes as convenient methods of generating well-separated message marks.

4.2.3 Example: Trellis Codes and Viterbi Decoding

To illustrate the idea of error correction codes, we now present an example of a *trellis code* (or *convolutional code*). Trellis codes are generally known to have good performance and serve as a foundation for some interesting research in watermarking [82, 190] (we briefly discuss this research in Chapter 5). In the following we describe the encoding and decoding processes for our example code. For more on the theory behind this type of coding, see [427].

Encoding

The encoder can be implemented as a state machine, as illustrated in Figure 4.8. The machine begins in state A and processes the bits of the input sequence one at a time. As it processes each input bit, it outputs 4 bits and changes state. If the input bit is a 0, it traverses the light arc coming from the current state and outputs the 4 bits with which that arc is labeled. If the input bit is a 1, it traverses the dark arc and outputs that arc's 4-bit label. Thus, a 4-bit message will be transformed into a 16-bit code word after encoding.

For example, consider encoding the 4-bit sequence 1010. Beginning at state A, we process the first (leftmost) bit in the sequence. The bold arc from state A is labled 0001, and those will be the first 4 bits of the coded word. The machine is now in state B when it processes the next bit, a 0. The light arc from state B is labeled 1100, and those will be the next 4 bits. Continuing on like this, we end up encoding our original 4-bit sequence with the 16-bit sequence

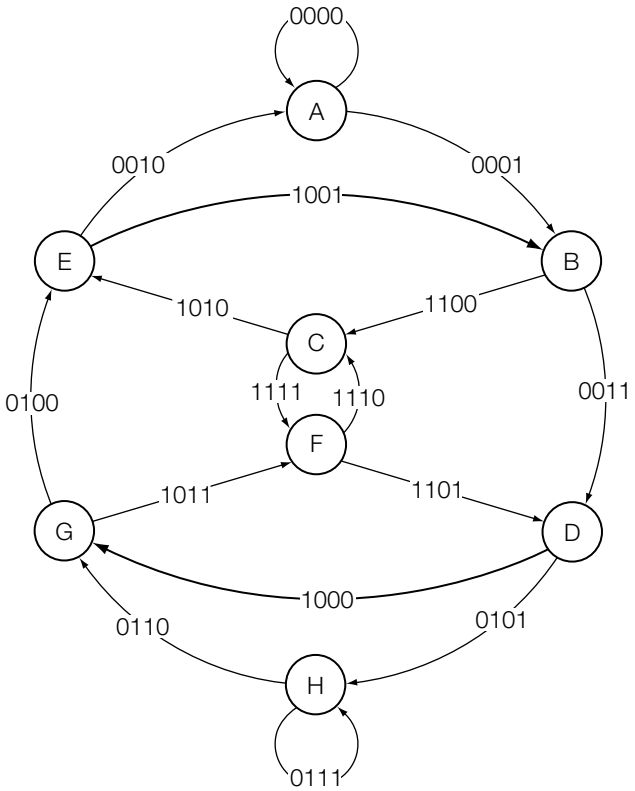


FIGURE 4.8

An eight-state convolutional code.

0001 1100 1111 1110. Note that each bit of the message affects not only the 4 bits used to encode it but also the encoding of several subsequent bits. Thus, the subsequent bits contain redundant information about earlier bits.

Figure 4.9 shows an alternative representation of the code, in a diagram called a *trellis*. Here, we have $8(L + 1)$ states, where L is the length of the input sequence. Each row of states corresponds to one state in Figure 4.8 at different times. Thus, state A0 in Figure 4.9 corresponds to state A in Figure 4.8 at the beginning of the encoding process, state A1 corresponds to state A after the first input bit has been processed, and so on. Each possible code word corresponds to a path through this trellis starting at state A0. Figure 4.9 highlights the code word for input 1010.

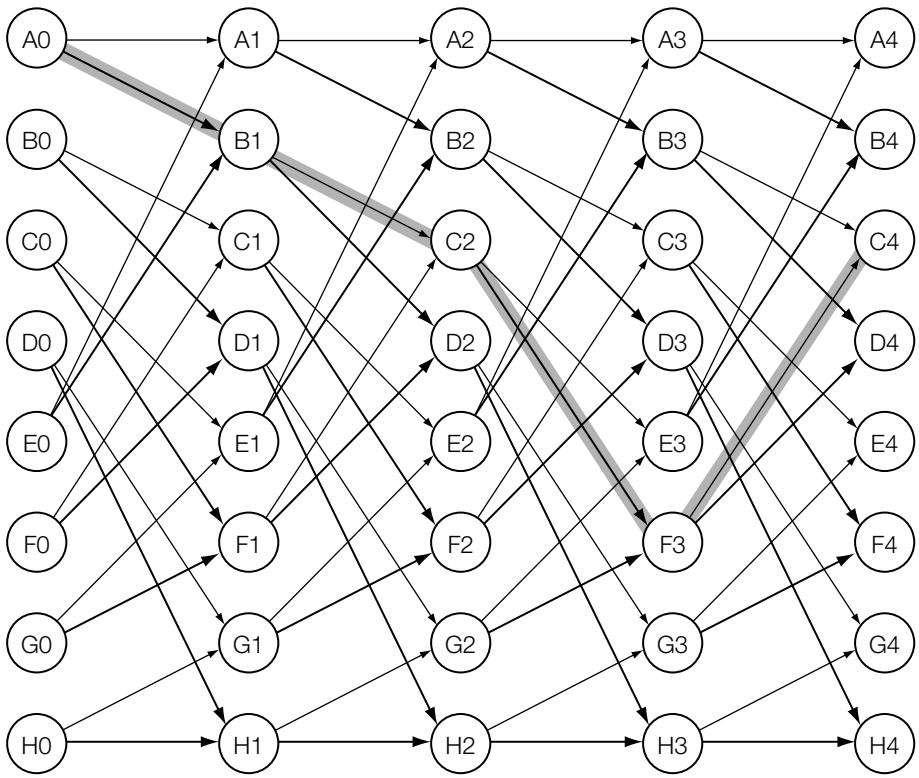


FIGURE 4.9

Trellis representation of the code in Figure 4.8. Each row corresponds to one of the eight states. Each column corresponds to an iteration of the encoding (column 0 is before encoding starts, column 1 is after encoding the first bit, and so on). The labels along the arcs have been removed for clarity. The highlighted path corresponds to the encoding of 1010.

The straightforward way of modulating code words generated by this code is to treat them as sequences of $4L$ bits, which are modulated in the same manner as in the E_SIMPLE_8/D_SIMPLE_8 system. Alternatively, we can replace the 4-bit arc labels with symbols drawn from a 16-symbol alphabet. This would lead us to assign a unique reference mark to each arc in the trellis of Figure 4.9 (one each for each of the 16 symbols in each of the L sequence locations). The message mark resulting from a given code word, then, is simply the sum of the reference marks for the arcs along the path that corresponds to that code word. This is known as *trellis-coded modulation* and is the approach we will take.

Decoding

Decoding a trellis-coded message is a matter of finding the most likely path through the trellis. In our case, the most likely path is the one that leads to the highest linear correlation or inner product between the received vector and the message vector for that path. This can be found efficiently using an algorithm known as a *Viterbi decoder*.

The Viterbi algorithm relies on the fact that the most likely path *through* each node in the trellis always includes the most likely path *up to* that node. Thus, once we find the most likely path from A0 to some node further into the trellis, we can forget about all the other possible paths from A0 to that node.

The algorithm proceeds by traversing the trellis from left to right. For each of the eight states in the columns of the trellis, it keeps track of the most likely path so far from A0 to that state, and the total inner product between the received vector and the reference marks along that path. In each iteration, it updates these paths and products. When it reaches the end of the trellis (i.e., the end of the coded message), it has found the most likely path from A0 to each of the eight final states. It is then a simple matter to decide which of the resulting eight paths is most likely.

We now describe the algorithm in more detail. To begin with, let us assume that the most likely path through the trellis is not restricted to the start point A0. We will introduce a simple method of enforcing this restriction later.

In this description, we use the following notation:

- $\mathbf{v}$ is the received vector being decoded.
- $\mathbf{w}_{i,j}$ is the reference mark associated with the arc in the trellis from state i to state j. For example, $\mathbf{w}_{\text{A0,B1}}$ is the reference mark associated with the arc from state A0 to B1.
- $\mathbf{p}[A \ldots H]$ is an array of eight paths. At any given time, $\mathbf{p}[i]$ is the most likely path that leads up to state i in the current column. For example, when processing column 3, $\mathbf{p}[C]$ is the most likely path that leads up to state C3.

- $\mathbf{z}[A \ldots H]$ is an array of eight inner products. At any given time, $\mathbf{z}[i]$ is the total product between the received vector, $\mathbf{v}$, and the reference marks for path $\mathbf{p}[i]$.

At the beginning of the algorithm, $\mathbf{p}[A \ldots H]$ are initialized to the null path, and $\mathbf{z}[A \ldots H]$ are initialized to 0. In the first iteration, we compute the inner product between the received vector and the 16 reference marks associated with the arcs that go from states in column 0 to states in column 1. To find the total product between the received vector and a path from a state in column 0 to a state in column 1, we add the product with the corresponding arc to the value in $\mathbf{z}$ that corresponds to the column 0 state. For example, the inner product for the path from A0 to B1 is just $\mathbf{z}[A] + \mathbf{v} \cdot \mathbf{w}_{\text{A0,B1}}$. Similarly, the product for the path from E0 to B1 is $\mathbf{z}[E] + \mathbf{v} \cdot \mathbf{w}_{\text{E0,B1}}$.

By comparing the inner products for the two paths that lead up to a state in column 1, we can decide which path to that state is most likely, and update $\mathbf{p}$ and $\mathbf{z}$ accordingly. Thus, for example, if

$$\mathbf{z}[A] + \mathbf{v} \cdot \mathbf{w}_{\text{A0,B1}} > \mathbf{z}[E] + \mathbf{v} \cdot \mathbf{w}_{\text{E0,B1}}, \tag{4.10}$$

then we would update

$$\mathbf{z}[B] \leftarrow \mathbf{z}[A] + \mathbf{v} \cdot \mathbf{w}_{\text{A0,B1}} \tag{4.11}$$

$$\mathbf{p}[B] \leftarrow \mathbf{p}[A] \text{ concatenated with the arc from A0 to B1.} \tag{4.12}$$

This process is repeated L times, until we reach the end of the trellis. At that point, we identify the most likely path by finding the highest value in $\mathbf{z}$. The path can then be converted into a decoded message.

To ensure that the most likely path always starts at A0, we can modify this algorithm by initializing only $\mathbf{z}[A]$ to 0 at the beginning. The other elements of $\mathbf{z}$ are initialized to an extreme negative value, indicating that the corresponding nodes have not yet been reached.

INVESTIGATION

8-Bit Trellis-Coded Watermarks

We now present a watermarking system, E_TRELLIS_8/D_TRELLIS_8, which employs the code of Figure 4.8 to embed 8-bit messages.

System 5: E_TRELLIS_8/D_TRELLIS_8

The E_TRELLIS_8 embedder takes an 8-bit message as input and maps it into a message pattern using trellis-coded modulation. One slight modification to the algorithm previously presented is that before modulating the message it appends two 0 bits onto the end, resulting in a 10-bit message to be modulated. This is done so that the code will provide some redundancy for the last bit in the message.

As in the E_SIMPLE_8 embedder, the message pattern, $\mathbf{w}_m$, is normalized to have zero mean and unit variance before embedding. This means that both systems have similar fidelity impacts when used with the same embedding strength parameter. The message pattern is embedded using blind embedding.

The D_TRELLIS_8 detector applies the Viterbi decoding algorithm described previously to map images into messages. Like the embedder, this detector is slightly modified for the extra two bits. Because we know, *a priori*, that these two bits must be 0, we restrict the decoder to ignore bold arcs during the last two iterations of decoding. In all other ways, the decoding algorithm is exactly as described previously.

The detector does not distinguish between marked and unmarked images. For every image, it outputs a sequence of 8 bits. If an image is not watermarked, the resulting bits are essentially random.

Experiments

To test this system, we ran the same experiment as was run on the E_SIMPLE_8/D_SIMPLE_8 system. Namely, 2,000 images were embedded with six different messages, resulting in 12,000 watermarked images. The embedding strength, again, was $\alpha = 2$.

Note that because we pad the message with two 0 bits there are 10 reference marks embedded rather than only 8. Thus, each mark is effectively embedded with an embedding strength of $\alpha/\sqrt{10}$.

In spite of the weaker embedding of individual reference marks, however, the performance of the E_TRELLIS_8/D_TRELLIS_8 system was significantly better than that of E_SIMPLE_8/D_SIMPLE_8. In all 12,000 watermarked images, only one message was incorrectly decoded, compared with 26 for the latter.

4.3 DETECTING MULTISYMBOL WATERMARKS

The two watermarking systems introduced previously are able to carry higher data payloads than those introduced in preceding chapters, but they lack the ability to distinguish between Works that contain watermarks and Works that do not. That is, these two detection algorithms map any Work into a sequence of 8 bits, regardless of whether that Work contains a watermark or not. We now turn our attention to the problem of deciding whether or not a multisymbol watermark is present in a Work.

In the case of direct message encoding, when the watermark consists of only one symbol, it is straightforward to determine whether a mark is present. The detector computes a detection value for each of the $|\mathcal{M}|$ messages. Only the highest of these is assumed to have any relevance to the embedded watermark. Thus, if the highest detection value is greater than a threshold, the detector can conclude that the corresponding watermark is present. The shape of

the resulting detection region depends only on the way the detection value is computed.

By contrast, in the case of a simple multisymbol system (such as the E_SIMPLE_8/D_SIMPLE_8 system described previously), the detector obtains several detection values that are assumed to reflect the presence of the individual symbols. By comparing different combinations of these values against a threshold, we can obtain different detection regions. Alternatively, we can apply a detection test that does not use the individual symbol detection values at all. A similar range of options is also open to us in more complex systems, such as E_TRELLIS_8/D_TRELLIS_8.

We discuss three basic approaches to the problem of determining presence of a multisymbol watermark. The first approach involves dividing the set of all possible messages into two sets: *valid* messages, which might be embedded by a watermark embedder, and *invalid* messages, which are never embedded. If a watermark detector detects an invalid message, it concludes that no watermark was embedded. The second approach uses linear correlation to decide whether all the symbols that encode the message were actually embedded. The third approach uses normalized correlation to determine whether the most likely message mark was actually embedded. Since the purpose of these techniques is to distinguish between watermarked and unwatermarked Works, we include a brief analysis of the false positive probability for each.

4.3.1 Detection by Looking for Valid Messages

A straightforward method of detecting whether or not a watermark is present in a Work is to declare that only a small subset of all possible symbol sequences represents legal messages.

Application

As a simple example, suppose we want to build a system that can embed 2^{16} different messages. We can represent each message with a sequence of 16 bits, and then append a 9-bit checksum obtained by adding the first 8 bits of the message to the second 8 bits of the message. This means that only 1 in every $2^9 = 512$ possible bit sequences corresponds to a legal message. In the detector, we decode all 25 bits, and then add the first 8 bits to the second 8 bits and compare the result with the last 9 bits. If the result matches, we say that the watermark is present.

This approach is commonly proposed for watermarking systems that have a high data payload [326, 404]. In fact, systems with very high payloads, such as 1,000 bits or more in a single Work (see Chapter 5), are often presented without discussion of how to determine whether a watermark is present [78, 82], since it can be assumed that the vast majority of possible messages will be defined as invalid by the application. For example, consider a watermark that is supposed to encode a passage of text as a string of 100 ASCII characters. Because most

possible strings of characters are not meaningful, it can reasonably be assumed that any Work that yields an intelligible string actually contains a watermark.

False Positive Probability

When watermarks are detected by distinguishing between valid and invalid messages, the false positive probability is easy to estimate. We begin by assuming that each message is as likely as any other to be found in an unwatermarked Work. This assumption is satisfied for a binary system if each bit has a 50% chance of being a 1, and if the bits are uncorrelated. In such a case, the probability of a false positive is simply the fraction of possible messages that is declared invalid.

Thus, in the previous example of 16-bit messages with 9-bit checksums, the false positive probability is 1/512. Clearly, if such an approach is to be reasonable for applications requiring very low false positive probabilities, the symbol sequences must be much longer, so that a much smaller fraction of them can be defined as valid.

4.3.2 Detection by Detecting Individual Symbols

An alternative method of detecting the presence of a length, L, multisymbol watermark is to test for each symbol independently, and report that the watermark is present only if every symbol's detection value exceeds a threshold.

Application

This approach is applicable when the detection values are linear correlations. However, as we shall see in the next section, it runs into trouble with normalized correlations.

When this strategy is applied in a system that uses linear correlation, the resulting detection regions are defined by the intersection of several hyperplanes. The four detection regions for a 2-bit binary system are illustrated in Figure 4.10. In this figure, the y-axis is the reference vector for bit 0, $\mathbf{w}_{r0}$, and the x-axis is the reference vector for bit 1, $\mathbf{w}_{r1}$. There are four possible messages—00, 01, 10, and 11—each of which has a distinct detection region. Any Work that lies outside all four detection regions is considered to contain no message.

False Positive Probability

To find the false positive probability, we begin with the probability that a given reference mark yields a correlation over the threshold. This can be estimated using the methods described in Chapter 7. We denote this probability as P_{fp0}. From P_{fp0}, we shall estimate the probability, P_{fp1}, that the most likely symbol at a given sequence location has a detection value higher than the threshold. Finally, we shall use P_{fp1} to estimate the probability that the most likely symbols in *all* of the sequence locations are over the threshold. This is the probability of a false positive, P_{fp}.

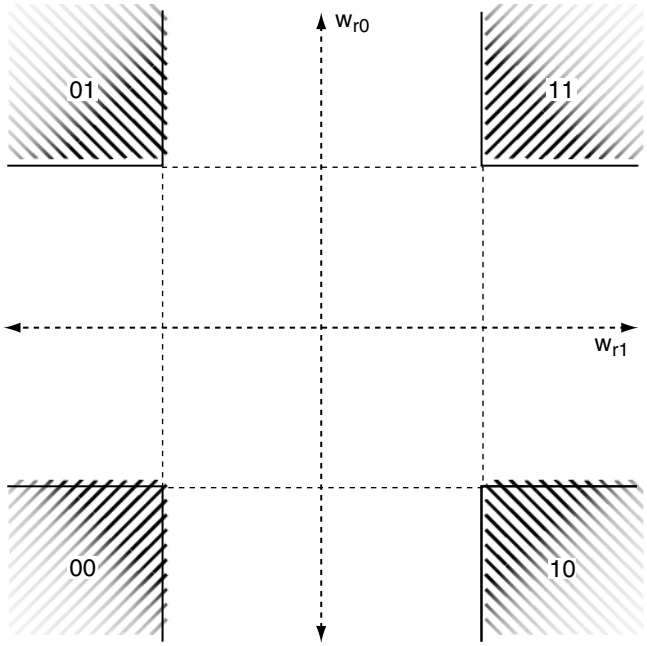


FIGURE 4.10

Detection regions for four possible messages encoded with two orthogonal watermark patterns. A watermark is detected only if the linear correlations between the Work and *both* patterns are above a threshold.

To estimate P_{fp1}, recall that for each of the L locations in the sequence of symbols being decoded the detector performs $|\mathcal{A}|$ correlations and selects the symbol that yields the highest value. It is this highest correlation value that will be compared against the threshold. Thus, P_{fp1} is the probability that at least one of $|\mathcal{A}|$ correlations exceeds the threshold when there is no watermark present. If we assume that the $|\mathcal{A}|$ marks for a given location are highly distinguishable, the probability, P_{fp1}, that at least one of them yields a correlation over the threshold is roughly equal to the sum of their separate probabilities; thus:

$$P_{\text{fp1}} \approx |\mathcal{A}| P_{\text{fp0}}. \tag{4.13}$$

From P_{fp1} we can now estimate the probability of a false positive. The detector only reports a detection if all L maximum correlations it finds are over the threshold. Assuming that the maximum correlation for each location is independent of that for all other locations, the probability, P_{fp}, that all of them will exceed the threshold, when running the detector on an unwatermarked Work, is given by

$$P_{\text{fp}} = P_{\text{fp1}}^{L} \approx \left(|\mathcal{A}| P_{\text{fp0}}\right)^{L}. \tag{4.14}$$

If the size of the alphabet is small enough, relative to the message sequence length, L, then P_{fp} is usually lower than P_{fp0}. This means that as the sequence length increases we can reduce the detection threshold and maintain the same overall false positive probability.

4.3.3 Detection by Comparing against Quantized Vectors

The strategy described in the preceding section relies on the fact that the presence of several orthogonal reference marks has no effect on each mark's linear correlation detection value. Unfortunately, if the detector uses normalized correlation instead of linear correlation, the strategy will not work because the addition of each new pattern always decreases the detection value for all others.

For example, consider two orthogonal reference marks, $\mathbf{w_{r1}}$ and $\mathbf{w_{r2}}$. The projected vector extracted from a Work with only $\mathbf{w_{r1}}$ embedded might be $\mathbf{v}_1 = \mathbf{v_o} + \mathbf{w_{r1}}$. Its normalized correlation with $\mathbf{w_{r1}}$ is

$$\begin{aligned} z_{nc}(\mathbf{v}_1, \mathbf{w_{r1}}) &= \frac{(\mathbf{v_o} + \mathbf{w_{r1}}) \cdot \mathbf{w_{r1}}}{|\mathbf{v_o} + \mathbf{w_{r1}}||\mathbf{w_{r1}}|} \\ &= \frac{\mathbf{v_o} \cdot \mathbf{w_{r1}} + \mathbf{w_{r1}} \cdot \mathbf{w_{r1}}}{|\mathbf{v_o} + \mathbf{w_{r1}}||\mathbf{w_{r1}}|} . \end{aligned} \tag{4.15}$$

Now, if we add in $\mathbf{w_{r2}}$ to obtain $\mathbf{v}_2 = \mathbf{v_o} + \mathbf{w_{r1}} + \mathbf{w_{r2}}$, and test for $\mathbf{w_{r1}}$ again, we have

$$\begin{aligned} z_{nc}(\mathbf{v}_2, \mathbf{w_{r1}}) &= \frac{(\mathbf{v_o} + \mathbf{w_r}1 + \mathbf{w_{r2}}) \cdot \mathbf{w_{r1}}}{|\mathbf{v_o} + \mathbf{w_{r1}} + \mathbf{w_{r2}}||\mathbf{w_{r1}}|} \\ &= \frac{\mathbf{v_o} \cdot \mathbf{w_{r1}} + \mathbf{w_{r1}} \cdot \mathbf{w_{r1}}}{|\mathbf{v_o} + \mathbf{w_{r1}} + \mathbf{w_{r2}}||\mathbf{w_{r1}}|} . \end{aligned} \tag{4.16}$$

The numerator is unaffected. However, because $\mathbf{w_{r1}}$ and $\mathbf{w_{r2}}$ are orthogonal, $|\mathbf{v_o} + \mathbf{w_{r1}} + \mathbf{w_{r2}}| > |\mathbf{v_o} + \mathbf{w_{r1}}|$, so $z_{nc}(\mathbf{v}_2, \mathbf{w_{r1}}) < z_{nc}(\mathbf{v}_1, \mathbf{w_{r1}})$. The amount by which $\mathbf{w_{r1}}$'s detection value is decreased depends on the magnitude with which $\mathbf{w_{r2}}$ is embedded. The stronger we make $\mathbf{w_{r2}}$, the weaker $\mathbf{w_{r1}}$ becomes.

Now suppose we compare both the detection values for $\mathbf{w_{r1}}$ and $\mathbf{w_{r2}}$ against a threshold to decide whether a watermark is present. Let $z_{nc}(\mathbf{v}, \mathbf{w_{r1}})$ and $z_{nc}(\mathbf{v}, \mathbf{w_{r2}})$ be the two detection values, respectively. We say a watermark is present if $z_{nc}(\mathbf{v}, \mathbf{w_{r1}}) > \tau_{nc}$ and $z_{nc}(\mathbf{v}, \mathbf{w_{r2}}) > \tau_{nc}$. This is equivalent to comparing only the lesser of the two against the threshold as follows:

$$\min(z_{nc}(\mathbf{v}, \mathbf{w_{r1}}), z_{nc}(\mathbf{v}, \mathbf{w_{r2}})) > \tau_{nc}. \tag{4.17}$$

Because increasing the detection value for one pattern decreases the value for the other, $\min(z_{nc}(\mathbf{v}, \mathbf{w_{r1}}), z_{nc}(\mathbf{v}, \mathbf{w_{r2}}))$ attains its highest value when $z_{nc}(\mathbf{v}, \mathbf{w_{r1}}) = z_{nc}(\mathbf{v}, \mathbf{w_{r2}})$. Thus, the highest possible value for $z_{min} = \min(z_{nc}(\mathbf{v}, \mathbf{w_{r1}}), z_{nc}(\mathbf{v}, \mathbf{w_{r2}}))$, obtained by setting $|\mathbf{w_{r1}}| = |\mathbf{w_{r2}}| = K$ and $\mathbf{v} = \mathbf{w_{r1}} + \mathbf{w_{r2}}$, is

$$\begin{aligned} z_{min} &= \frac{\mathbf{w_{r1}} \cdot \mathbf{w_{r1}}}{|\mathbf{w_{r1}} + \mathbf{w_{r2}}||\mathbf{w_{r1}}|} \\ &= \frac{K^2}{(\sqrt{2}K)K} \\ &= \frac{1}{\sqrt{2}}. \end{aligned} \tag{4.18}$$

Thus, if we set our threshold higher than $1/\sqrt{2}$, we cannot possibly obtain any detections. In general, if we are embedding L orthogonal patterns, the best we can expect for the minimum of all L normalized correlations is $1/\sqrt{L}$.

A geometric interpretation of this limitation is shown in Figure 4.11. This shows the four individual detection regions for the four reference marks employed in a binary (i.e., 2-ary) system, such as that shown in Figure 4.10. The threshold has been set high enough that no two detection regions overlap, which means that no Works can be detected as containing more than one of the marks. Embedding two marks at equal strength in this case would make them both undetectable.

To solve this problem, we can use a different test for presence of the watermark. Rather than try to determine whether the mark is present on a symbol-by-symbol basis, we first identify the most likely message mark, and then test for that message mark as a whole. Identifying the most likely message mark amounts to a form of vector quantization.

Application

A general method for finding the most likely message mark and testing for its presence begins by decoding the extracted vector, $\mathbf{v}$, in the usual way. This gives a sequence of symbols, $m[1] \ldots m[L]$. At this point, we reencode the decoded message to obtain a single message mark. In the simplest case, the message mark obtained is just the summation of the most likely reference marks. In more sophisticated systems, the decoding and reencoding steps might include application of error correction codes.

The reencoding gives us the message mark, $\mathbf{w}_m$, which would be embedded in a Work to embed the message m. We now apply a test for presence of $\mathbf{w}_m$, as if it were the only possible watermark in the system. If we wish to use normalized correlation detection, we can compute the normalized

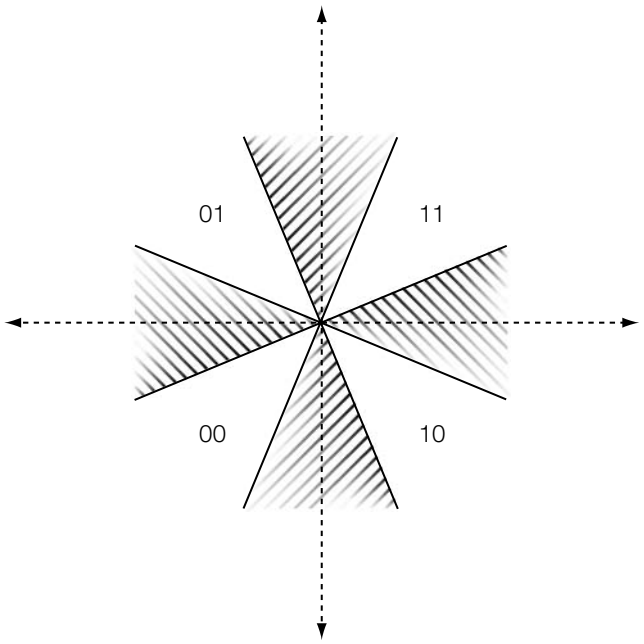


FIGURE 4.11

A geometric interpretation of the upper bound on normalized correlations for four reference marks in a 2-ary system.

correlation between the extracted vector, $\mathbf{v}$, and $\mathbf{w}_m$, and compare the resulting value against a threshold. This results in the detection regions shown in Figure 4.12.

False Positive Probability

In general, the false positive probability is bounded from above by the number of possible messages, $|\mathcal{M}|$, times the probability that any one message will yield a false positive, P_{fp0}. This bound is tight when the detection regions for the different messages do not overlap.

For example, in a system that uses normalized correlation as a detection measure, the detection region for each message is an N-cone (see Chapter 3). If the angle subtended by these cones is less than the minimum angle between two message vectors, none of the cones overlap, and the probability of obtaining a false positive with the entire system is exactly the sum of the false positive probabilities for the $|\mathcal{M}|$ separate messages. Assuming that these probabilities are all the same, P_{fp0}, the probability of false positives is simply $P_{\text{fp}} = |\mathcal{M}|P_{\text{fp0}}$.

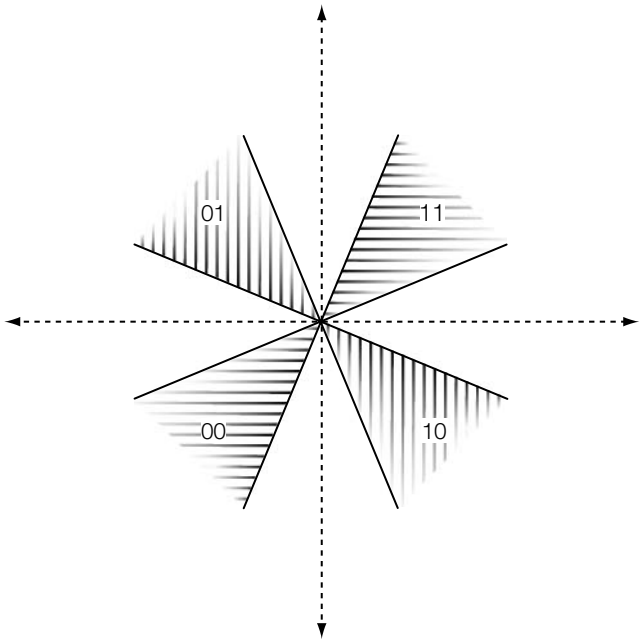


FIGURE 4.12

Illustration of the reencoding method for four reference marks.

INVESTIGATION

Block-Based 8-Bit Watermarking with Normalized Correlation Detection

We illustrate the method of Figure 4.12 by presenting a modified version of the E_BLK_BLIND/D_BLK_CC system. The system presented here embeds and detects 8-bit watermarks, and uses the correlation coefficient to determine whether or not a watermark is present.

System 6: E_BLK_8/D_BLK_8

The E_BLK_8 embedder is almost identical to E_BLK_BLIND, except that rather than taking a 1-bit message and an 8×8 reference mark as input it takes an 8-bit message, m, and a seed for a pseudo-random number generator. The message is encoded with the trellis code of Figure 4.9. As in the E_TRELLIS_8 algorithm described previously, two 0s are appended to the message before encoding, to add some redundancy for the last few bits. This results in a sequence of 10 symbols drawn from a 16-symbol alphabet.

The symbol sequence is then modulated using code division in the same way as in E_TRELLIS_8, except that 8 × 8 reference marks are used instead of full-size reference patterns. Finally, the resulting message mark, $\mathbf{w}_m$, is embedded in the same way as in E_BLK_BLIND.

The D_BLK_8 detector begins by extracting a vector, $\mathbf{v}$, in the same manner as D_BLK_CC. The extracted vector is then decoded using a Viterbi decoder, to obtain the received message, m. This is the message that is most likely to be in the image.

To determine whether or not the message actually *is* in the image, the detector then *reencodes* m, using the same encoding used by the embedder. This yields the message mark, $\mathbf{w}_m$, that is most likely to have been embedded. If the normalized correlation between $\mathbf{w}_m$ and the extracted vector, $\mathbf{v}$, is greater than a threshold, τ_{cc}, the detector concludes that the watermark is present and it outputs m. Otherwise, it outputs *no watermark*.

Experiments

This system was tested on our usual set of 2,000 images. Each image was embedded with the messages $m = 0$, 255, 101, 154, 128, and 127, resulting in a total of 12,000 watermarked images. The embedding strength was $\alpha = 2$. The detection threshold was $\tau_{cc} = 0.65$, which leads to an estimated false positive probability of about 10^{-6}.

The seed for the random number generator was carefully chosen. As with the other systems in this chapter, because the reference marks are generated pseudo-randomly, different seeds result in different amounts of code separation. In the E_SIMPLE_8/D_SIMPLE_8 and E_TRELLIS_8/D_TRELLIS_8 systems, this is not a serious problem because, in the high dimensionality of media space, the variation in code separation is small. However, in the E_BLK_8/D_BLK_8 system, marking space has only 63 dimensions (64 dimensions minus 1 for zeroing the means). Because of this low dimensionality, the variation in the amount of code separation is more severe.

Because we are encoding only 8 bits, it is possible to enumerate all 256 message vectors that result from a given seed, and correlate each of them against all others. The maximum correlation found gives an idea of how good the seed is. Good seeds lead to low maximum correlations. In this way, we can try several different seeds and choose the one that leads to the best code. The seeds we tested led to maximum correlations between about 0.73 and 0.77. For our experiments, we used a seed that led to 0.73.

Figure 4.13 shows the distribution of detection values obtained. The dashed line indicates detection values obtained when the D_BLK_8 detector was run on 2,000 unwatermarked images. The solid line indicates the detection values obtained on the 12,000 watermarked images. There were no false positives detected. The E_BLK_8 embedder failed to embed in only 109 of the trials, giving it a measured effectiveness of over 99%.

As shown in Figure 4.13, the detection values obtained from unwatermarked images are considerably higher than those obtained with our earlier, 1-bit watermarking systems. This is because each reported detection value is effectively the maximum of 256 possible values. That is, the D_BLK_8 detection algorithm obtains the same value that would be obtained from a detector that exhaustively correlated against each of the 256 possible message vectors and returned the highest value.

Note that because the detection threshold ($\tau_{cc} = 0.65$) is less than the maximum correlation between message marks (0.73) there is some risk of obtaining bit errors. That is, the detection cones for at least two different messages overlap. If the embedder moves a Work into the cone for one of these messages, the result might actually be closer to the other message, thus resulting in a different message being detected. However, this did not occur in our experiment. Out of the 12,000 messages embedded, 16 were not correctly decoded, but in each of those cases the correlation coefficient was below 0.65. Therefore, the detector output *no watermark* rather than an incorrect message.

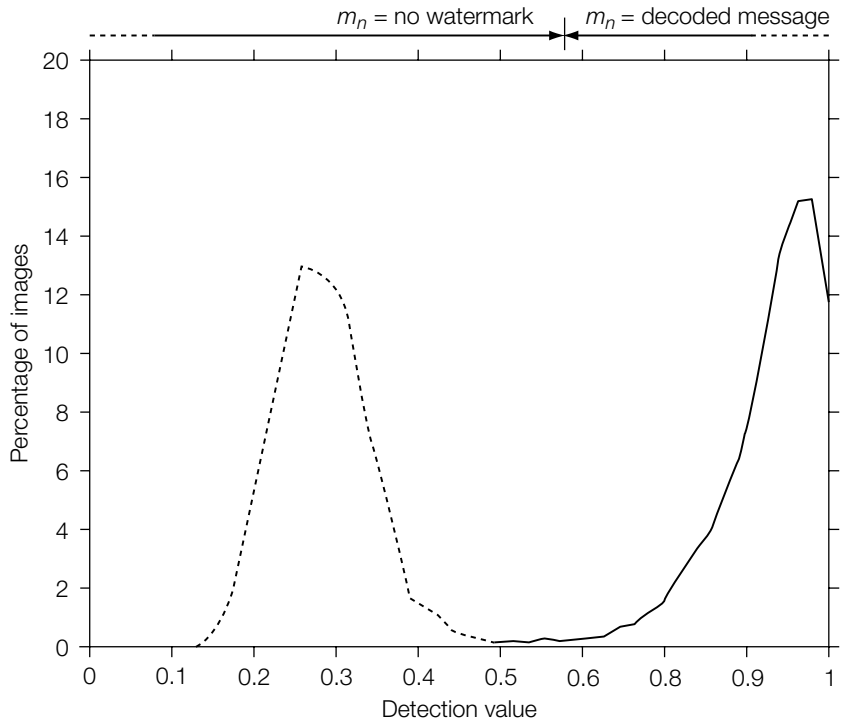


FIGURE 4.13

Test results for the E_BLK_8/D_BLK_8 watermarking system.

4.4 SUMMARY

This chapter has discussed basic methods for mapping messages into watermarks. Our focus here has been on simple methods of modifying the types of watermarking systems presented as examples in earlier chapters. The following points were made.

- Message coding is often implemented in two steps:
  - *Source coding* maps messages into sequences of symbols.
  - *Modulation* maps sequences of symbols into message marks that can be embedded in content.
- The key issue in the design of a code is *code separation* (i.e., the distances between different message marks in marking space). If these distances are too small, there will be a high chance that one message embedded in a Work will be detected as a different message. In the case of correlation-based watermarks, the distance of interest is the correlation between message marks.
- Randomly generated codes, in which each message maps directly into a message mark, are likely to exhibit good behavior.
  - If the number of messages is large compared to the dimensionality of marking space, the average code separation of a random code is likely to be close to optimal.
  - If the number of messages is small compared to the dimensionality of marking space, random message marks are likely to have a correlation close to zero.
  - Zero correlation between marks can be useful if we wish to embed more than one distinct message in a linear correlation watermarking system.
- Multisymbol codes, in which messages are first mapped into sequences of symbols (source coding) and then systematically mapped into message marks (modulation), reduce computational complexity.
- Modulation can be performed in a variety of ways, including the following:
  - *Time-division multiplexing:* Dividing a Work in the temporal domain and embedding one symbol in each segment.
  - *Space-division multiplexing:* Dividing a Work in the spatial domain and embedding one symbol in each segment.
  - *Frequency-division multiplexing:* Dividing a Work in the frequency domain and embedding one symbol in each segment.

- *Code-division multiplexing:* Using a different reference mark for each possible combination of symbol and sequence location.

- Error correction coding (ECC) increases the code separation of multi-symbol codes, bringing their performance closer to that of random codes. An example of a trellis code with a Viterbi decoder was presented.

- There are several ways a detector can determine whether or not a multi-symbol message is present in a Work. These include the following:

  - Define a large fraction of possible messages to be "invalid" by, for example, appending a checksum to the end of each embedded message. Any message with an incorrect checksum is invalid. If a valid message is detected, the detector declares that the watermark is present.

  - Apply a separate detection test to each symbol in the message. If all the symbols are detected, the detector declares that the watermark is present.

  - Identify the most likely message mark in the Work. A simple, general way to do this is to demodulate and decode the most likely message, and then reencode and remodulate that message to obtain the most likely message mark. Once the most likely mark is identified, apply a separate test for its presence in the Work, such as computing the correlation coefficient between the mark and an extracted vector.

This page intentionally left blank

CHAPTER

5 Watermarking with Side Information

The watermarking systems presented in Chapter 4 all fit into the basic model shown in Figure 5.1. In each embedder, a message, m, is first encoded as a message mark, $\mathbf{w}_m$. This is then modified by simple scaling to obtain an added mark, $\mathbf{w_a}$, which is added to the cover Work. The detectors are blind, in that they are not given any information about the original, unmarked Work. The embedders in these systems can be thought of as blind in the same sense. They ignore the available information about the cover Work during both the coding of the message (*blind coding*) and the modification of the message mark in preparation for embedding (*blind embedding*).

In contrast, in Chapter 3, we discussed two types of systems that cannot be fit into Figure 5.1. The first type is a system that uses *informed detection* (see Section 3.3.1 of Chapter 3), in which the detector is provided with information about the original, unmarked cover Work. It can then subtract this information from the received Work, eliminating any interference between the cover Work and the watermark. With an informed detector, then, the detectability of the watermark is affected only by the distortions to which the watermarked Work is subjected after embedding; it is not affected by the cover Work itself.

The second type of nonblind system discussed in Chapter 3 is a system that uses *informed embedding* (see Section 3.3.2 of Chapter 3). Here, the embedder examines the cover Work during the modification of the message mark in preparation for embedding. This was illustrated with the E_FIXED_LC/D_LC watermarking system (System 2), in which the message mark is scaled so that we obtain a fixed linear correlation detection value. Using this technique, we found it is possible to obtain virtually 100% effectiveness.

In this chapter, we first examine the idea of informed embedding in more detail. Specifically, we look at systems that fit into the model of Figure 5.2. In this context, embedding can be cast as an optimization problem. We are either trying to maximize the robustness of the watermark while maintaining a fixed fidelity or trying to maximize the fidelity while maintaining a fixed robustness. Implementing either of these strategies requires a means of measuring fidelity and robustness.

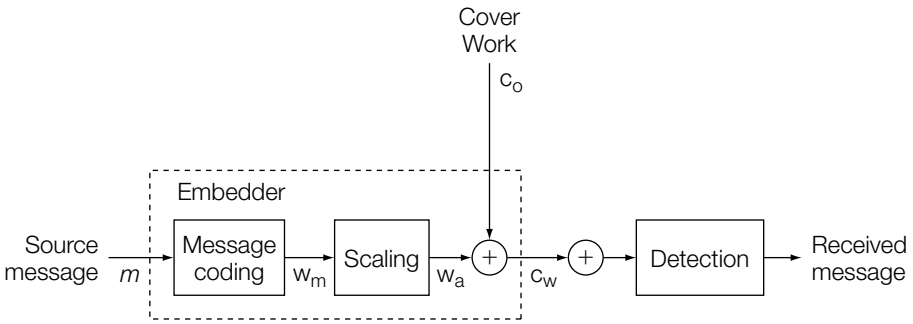


FIGURE 5.1

Watermarking with blind embedders.

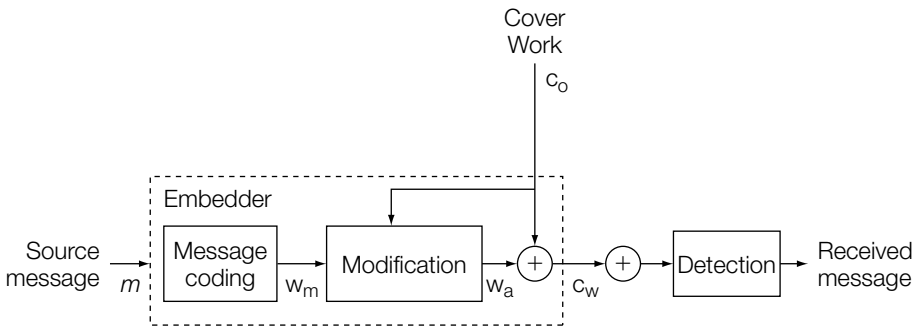


FIGURE 5.2

Watermarking with informed embedding.

Historically, the earliest examples of informed embedding used only perceptual models. These systems locally amplified or attenuated the added watermark pattern to improve fidelity. Such systems are discussed in Chapter 8. However, these early systems do not employ any explicit consideration of the detection region, and the perceptual shaping is treated as a form of noise.

In Section 5.1 we concentrate on the issue of estimating robustness and use a simple mean square error (MSE) for fidelity. We develop techniques for performing informed embedding when the detector uses normalized correlation rather than linear correlation, and argue that in this case the detection statistic itself is not a good measure of robustness.

Section 5.2 then introduces the idea of *informed coding*, in which the problem of communication using watermarking is viewed as a joint problem for the encoder and decoder. We introduce a formal model of watermarking and formulate the notion of an *achievable region*, providing a precise meaning of *optimal* watermarking. As a preparation for the main optimality result of Section 5.3, the remainder of the section is devoted to an overview of some classical tools in

communication theory. In particular, we present a geometrical interpretation of Gaussian signals and sketch a geometrical proof of Shannon's theorem on communicating over an additive white Gaussian noise (AWGN) channel.

Section 5.3 then introduces the idea of *informed coding*, in which the encoder uses knowledge of the *actual* cover Work as side information to achieve an *overall* optimal performance. This contrasts with the methods previously discussed in which only *statistical* knowledge of the host cover Work is used. In this general model we will consider the watermark encoder as a black box that produces a marked Work given an original Work and a message m. In particular, we will no longer assume any internal structure adding some signal $\mathbf{w}$ to a host signal $\mathbf{c_o}$.

The general approach to informed coding is to use what we term *dirty-paper coding* (see Section 5.3), in which each message can be represented by several dissimilar marked Works. The embedder searches through the code book of potential marked Works for a given message and selects the one that is closest to the cover Work. The optimal dirty-paper method exploits a large code book C of potential marked Works and uses a technique referred to as *distortion compensation* to efficiently search this set of Works. This is discussed in Section 5.3.2. The decoder is able to retrieve the embedded message by comparing the received (and potentially degraded) Work against the shared code book C. Most surprisingly, it can be proven that watermarking with *only* side information at the encoder can perform as well as systems that use side information at both the encoder and decoder. This result is stated in the famous "Writing on Dirty Paper" theorem by Max Costa [88] (Theorem 6), a sketch proof of which is provided.

The theoretical analysis of dirty-paper codes does not tell us how to build them in practice. In general, the analysis refers only to random codes, which are only practical to implement for very small data payloads. We therefore conclude this chapter, in Sections 5.3.3 and 5.3.4, with a discussion of some preliminary attempts to realize the theoretical promise of dirty-paper codes with more practical systems. Chapter 6 provides further illustrations of practical systems.

5.1 INFORMED EMBEDDING

The basic idea of informed embedding was introduced in Chapter 3 and illustrated with the E_FIXED_LC/D_LC watermarking system. In an informed embedder, the pattern, $\mathbf{w_a}$, added to a Work depends on the message mark, $\mathbf{w}_m$, and the Work itself.

In this section we more closely examine the problem of designing informed embedders. We begin by looking at how the design of an informed embedder can be cast as an optimization problem. We then examine two different methods for assessing the strength of embedded watermarks: first, by looking

at the detection measure itself, and second, by defining a different measure that attempts to predict the robustness of the embedded mark. We conclude that although the former approach is fine for simple, linear correlation systems, the latter approach is required for normalized correlation.

5.1.1 Embedding as an Optimization Problem

In the absence of a fidelity constraint (i.e., no need to preserve the cover Work), the optimal method of informed embedding is trivial. Consider a simple watermarking system that operates directly in media space. The optimal strategy for choosing the added pattern, without concern for fidelity, would be to subtract the cover Work from the message pattern. Thus, $\mathbf{w_a} = \mathbf{w}_m - \mathbf{c_o}$. This subtraction has been referred to as *precancellation* [75, 319]. When $\mathbf{w_a}$ is added to the cover Work to obtain the watermarked Work, the result is just $\mathbf{c_w} = \mathbf{w}_m$. The detectability of the watermark would then depend only on the distortions subsequently applied to $\mathbf{c_w}$.

For most watermarking systems, the "optimal" system, described previously, would be unacceptable, in that it simply replaces the cover Work with the message pattern and thus results in an output signal that does not resemble the original Work at all. However, in systems that project Works into a marking space much smaller than media space, the approach is occasionally feasible. Recall that in such systems we embed by first extracting a mark, $\mathbf{v_o}$, from the Work, and then choosing an added mark, $\mathbf{v_a}$, and adding it to $\mathbf{v_o}$ to obtain the watermarked vector, $\mathbf{v_w}$. The Work is then modified so that it yields $\mathbf{v_w}$ when the extraction process is applied. It is sometimes possible to modify Works so that they yield any arbitrary vector, while still maintaining fidelity. This allows us to let $\mathbf{v_w}$ equal the message mark, $\mathbf{v}_m$. Nevertheless, in most cases, $\mathbf{v_w}$ must lie somewhere between the message mark and the original extracted mark, $\mathbf{v_o}$.

When fidelity requirements prevent us from using the simple approach of setting $\mathbf{v_a} = \mathbf{v}_m - \mathbf{v_o}$, the problem of finding the best added mark, $\mathbf{v_a}$, can be cast as an optimization problem. The specific optimization problem we wish to solve depends on the application. There are two main possibilities:

- Find the added mark that maximizes robustness while keeping within a prescribed limit on perceptual distortion.
- Find the added mark that minimizes perceptual distortion while maintaining a prescribed level of robustness (the approach used in the E_FIXED_LC embedder).

More complicated expressions of the problem, where we try to balance the tradeoff between fidelity and robustness, are also possible. For example, we may specify a maximum level of distortion *and* a minimum level of robustness. If the watermark cannot be embedded within both constraints, one or the other should be automatically relaxed. Thus, the embedder might try to maximize

robustness while staying within the maximum distortion. However, if the highest robustness attainable is lower than a specified minimum, the perceptual constraint is relaxed.

To implement any of these strategies, we need methods for measuring perceptual distortion and robustness. Methods for measuring perceptual distortion are discussed in Chapter 8. Methods for measuring robustness are discussed in the following.

5.1.2 Optimizing with Respect to a Detection Statistic

The simplest way to measure robustness is to assume that a watermark with a high detection value is more robust than one with a low detection value, so that the detection statistic itself becomes our measure of robustness. Thus, when embedding a watermark with fixed robustness, the user specifies a desired detection value, and the embedder tries to achieve that detection value with the minimum possible distortion. For example, in the E_FIXED_LC embedder, we specify a desired linear correlation as the sum of the detection threshold, τ_{lc}, and a "strength" parameter, β.

The assumption that high detection values indicate high robustness is generally true for linear correlation. A high detection value means the watermarked vector is far from the detection threshold, and more distortion of any type can be applied before the watermark is undetectable. However, this assumption is *not* true for many other detection statistics, including normalized correlation. To see that the assumption can fail, we now develop an algorithm for embedding watermarks with a fixed normalized correlation, and show experimentally that higher normalized correlations do not always result in more robust watermarks.

The experiment we wish to perform requires a new embedding algorithm, because the E_FIXED_LC algorithm is inappropriate for normalized correlation. The reason for this is illustrated in Figure 5.3. This figure shows the effect of the embedding algorithm on several unmarked Works. Each open circle represents the location in marking space of an unmarked Work and is connected by an arrow to a filled circle that represents the corresponding watermarked Work. The E_FIXED_LC algorithm ensures that the linear correlation between each watermarked Work and the watermark vector is constant, which means that they all lie in a narrow embedding region. The shaded area shows the detection region for a normalized correlation detector. Because the embedding region is not completely contained within the detection region, it is clear that the E_FIXED_LC embedder will fail to embed the watermark in some Works. We therefore need an embedding algorithm that creates a different embedding region, one that does lie completely within the detection region.

The solution we test here is to fix the normalized correlation at some desired value, a_{nc}, rather than fixing the linear correlation. For each Work, the embedder finds the closest point in marking space that has the desired normalized correlation

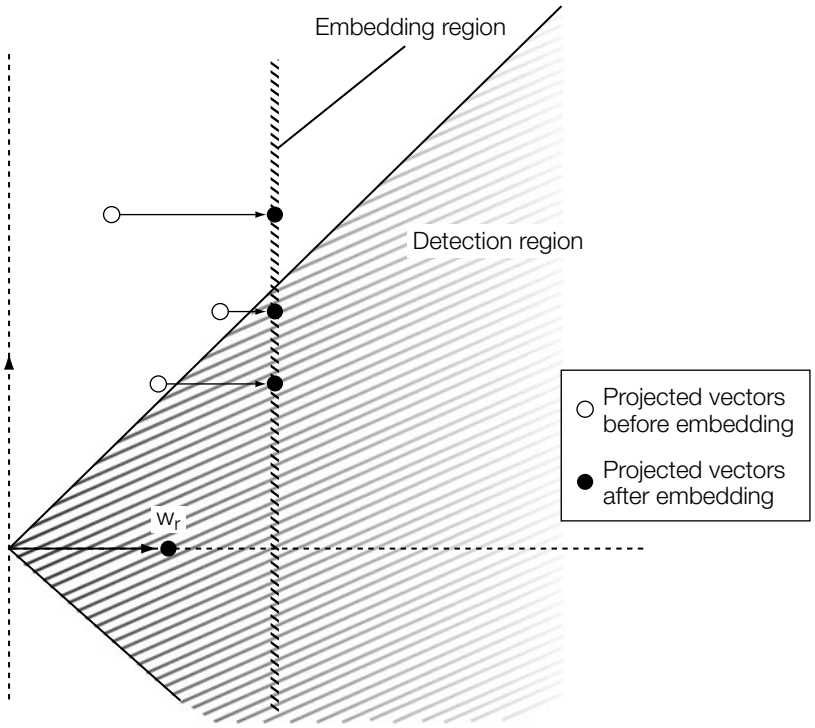


FIGURE 5.3

Illustration of the problem associated with a fixed linear correlation embedding strategy. The figure shows the embedding region for a fixed linear correlation embedder and the detection region for a normalized correlation detector. Because the embedding region is not entirely contained in the detection region, some Works will fail to be watermarked.

with the watermark. If we use Euclidian distance to determine proximity, this approach would lead to the embedding behavior illustrated in Figure 5.4.

Implementing the fixed normalized correlation embedder is a simple matter of geometry. The embedding region is the surface of a cone, centered on the reference vector, $\mathbf{w_r}$. Therefore, we are trying to find the closest point on a cone to a given point, $\mathbf{v_o}$. Clearly, this point will lie in the plane that contains both $\mathbf{w_r}$ and $\mathbf{v_o}$. Thus, we begin by reducing the problem to two dimensions by using Gram-Schmidt orthonormalization to find two orthogonal coordinate axes for this plane:

$$\mathbf{X} = \frac{\mathbf{w_r}}{|\mathbf{w_r}|} \tag{5.1}$$

$$\mathbf{Y} = \frac{\mathbf{v_o} - \mathbf{X}(\mathbf{v_o} \cdot \mathbf{X})}{|\mathbf{v_o} - \mathbf{X}(\mathbf{v_o} \cdot \mathbf{X})|}. \tag{5.2}$$

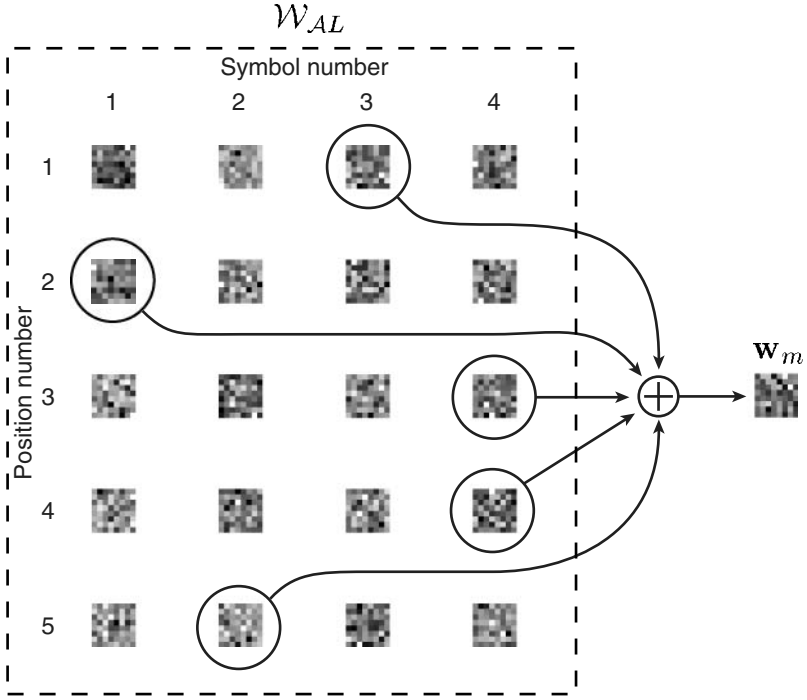


FIGURE 5.4

Illustration of the problem associated with a fixed linear correlation embedding strategy. The figure shows the embedding region for a fixed linear correlation embedded and the detection region for a normalized correlation detector. Because the embedding region is not entirely contained in the detection region, some Works will fail to be watermarked.

Each point on the plane can be expressed with an x and y coordinate as $x\mathbf{X} + y\mathbf{Y}$. The x-axis is aligned with the reference vector. The coordinates of $\mathbf{v_o}$ are

$$x_{\mathbf{v_o}} = \mathbf{v_o} \cdot \mathbf{X} \tag{5.3}$$

$$y_{\mathbf{v_o}} = \mathbf{v_o} \cdot \mathbf{Y}. \tag{5.4}$$

Note that $y_{\mathbf{v_o}}$ is guaranteed to be positive.

The embedding region intersects with the $\mathbf{XY}$ plane along two lines, as shown in Figure 5.4. Because $y_{\mathbf{v_o}}$ is positive, the closest point will be on the upper line. This line is described by the vector x_t, y_t, with $x_t = \cos(\theta)$ and $y_t = \sin(\theta)$, where θ is half the angle subtended by the cone of the embedding region. Because $\theta = \cos^{-1}(t_{\text{nc}})$, where t_{nc} is the desired normalized correlation, we have

$$x_t = t_{\text{nc}} \tag{5.5}$$

$$y_t = \sqrt{1 - t_{\text{nc}}^2}. \tag{5.6}$$

The closest point to $\mathbf{v_o}$ along the line described by x_t, y_t is

$$x_{\mathbf{v_w}} = x_t(x_t x_{\mathbf{v_o}} + y_t y_{\mathbf{v_o}}) \tag{5.7}$$

$$y_{\mathbf{v_w}} = y_t(x_t x_{\mathbf{v_o}} + y_t y_{\mathbf{v_o}}). \tag{5.8}$$

Finally, the watermarked version of $\mathbf{v_o}$ is given by

$$\mathbf{v_w} = x_{\mathbf{v_w}} X + y_{\mathbf{v_w}} Y. \tag{5.9}$$

Using Equations 5.2 through 5.9, we can now construct an informed embedder for a system that uses normalized correlation as its detection statistic.

INVESTIGATION

Fragility of a Fixed Normalized Correlation Strategy for Informed Embedding

We now modify the E_BLK_BLIND/D_BLK_CC (System 3) watermarking system to embed with a fixed normalized correlation. As in that system, the system developed here embeds 1 bit of information, either $m = 0$ or $m = 1$. The target normalized correlation is specified as the sum of the detection threshold, τ_{lc}, that will be used in the detector and a "strength" parameter, β.

This investigation will reveal that the robustness of watermarks to additive white noise can actually *decrease* with increased β. The reason for this is explained at the end of the investigation.

System 7: E_BLK_FIXED_CC/D_BLK_CC

The basic steps of the E_BLK_BLIND algorithm are:

1. Use the extraction function, $X(\cdot)$, to extract a vector, $\mathbf{v_o}$, from the unmarked image, $\mathbf{c_o}$.
2. Compute a new vector in marking space, $\mathbf{v_w}$, that is acceptably close to $\mathbf{v_o}$ but lies within the detection region for the desired watermark, $\mathbf{w_r}$.
3. Perform the inverse of the extraction operation on $\mathbf{v_w}$ to obtain the watermarked image, $\mathbf{c_w}$.

The details of the extraction and inverse extraction processes are described in Chapter 3 and will not be changed here. Instead, we concentrate only on Step 2, computing $\mathbf{v_w}$ according to the method previously described.

As in the E_BLK_BLIND embedding algorithm, we encode a 1-bit message, m, by determining the sign of a given 8×8 reference mark, $\mathbf{w_r}$:

$$\mathbf{w}_m = \begin{cases} \mathbf{w_r} & \text{if } m = 1 \\ -\mathbf{w_r} & \text{if } m = 0. \end{cases} \tag{5.10}$$

With a target normalized correlation of $t_{cc} = \tau_{cc} + \beta$, an extracted mark of $\mathbf{v_o} = X(\mathbf{c_o})$, and the given message mark, $\mathbf{w}_m$, we can directly apply Equations 5.2 through 5.9 to obtain the new vector, $\mathbf{v_w}$, for Step 2. Step 3 then proceeds exactly as in E_BLK_BLIND, to obtain the watermarked image.

Experiments

As with other watermarking systems, we first test the performance of the E_BLK_FIXED_CC/D_BLK_CC system with no distortions applied between embedding and detection. Two thousand images were embedded with messages of $m = 0$ and $m = 1$, using two different values of β, resulting in 4,000 watermarked images. The detection threshold was $\tau_{cc} = 0.55$. The two values of β were $\beta = 0.25$ and $\beta = 0.40$. Figures 5.5(a) and (b) show the results. As expected, the normalized correlations obtained from E_BLK_FIXED_CC fall into a very narrow range, and a higher value of β increases the average.

We would hope that increasing β should increase robustness. To test this, we applied a simple attack to all the watermarked images by adding white Gaussian noise, with a standard deviation of 10. We then ran the D_BLK_CC detector on the distorted images, using the same threshold value as was used during embedding, $\tau_{cc} = 0.55$. In the case of $\beta = 0.25$, about 85% of the watermarks were correctly detected after the white noise attack. Increasing β to 0.40, however, actually *decreased* the percentage of surviving watermarks to 66%. Thus, the higher value of β yielded *worse* robustness than did the lower value.

These results indicate that, unlike with linear correlation, higher normalized correlations do not necessarily lead to higher robustness. This can be explained by realizing that, whereas linear correlation measures the magnitude of the signal, normalized correlation essentially measures the signal-to-noise ratio (SNR). The SNR can be high even if the signal is very weak, as long as the noise is proportionally weak. However, if we add a fixed amount of noise to a weak signal, the resulting SNR plummets.

This is exactly what happened in our experiment. Figure 5.6 gives a geometric illustration of the problem. With a high value of β, the E_BLK_FIXED_CC embedder yields a vector close to the origin of the cone. Perturbing this vector in almost any direction is likely to send it outside the detection region. A low value of β yields a vector deeper within the cone, and thus is more resilient to noise.

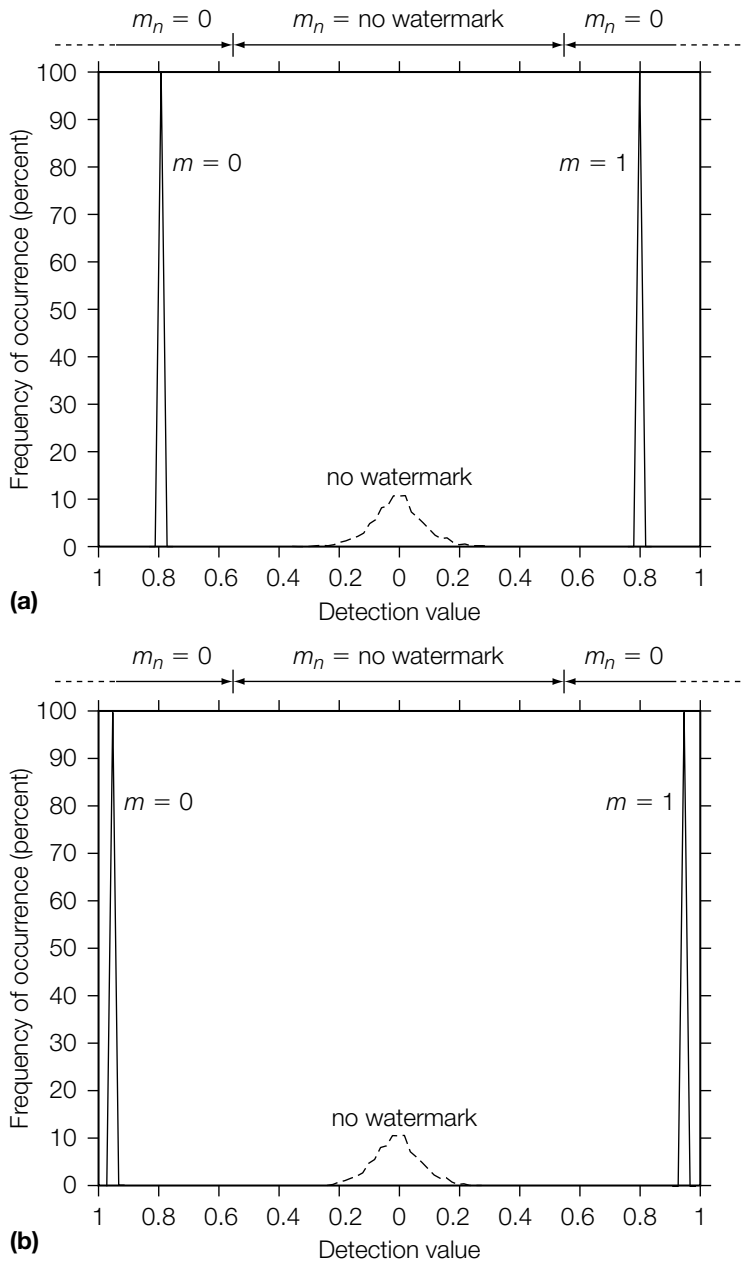


FIGURE 5.5

Distribution of detection values for two different values of β in the E_BLK_FIXED_CC/D_BLK_CC system. The strength parameter for these experiments was (a) β = 0.25 and (b) β = 0.40.

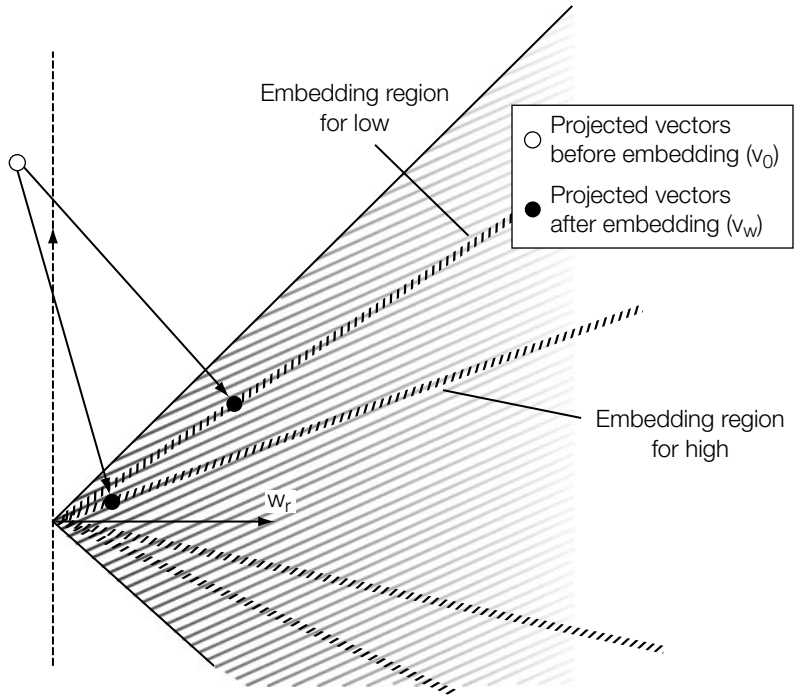


FIGURE 5.6

Illustration of the problems associated with a fixed normalized correlation embedding strategy.

5.1.3 Optimizing with Respect to an Estimate of Robustness

The problem with the E_BLK_FIXED_CC algorithm is that its "strength" parameter, β, is based on the wrong measure. Normalized correlation does not measure robustness. Thus, we need to express our optimization problem with respect to a different measure, one that is more directly related to robustness.

The formula used for such a measure is dependent on the method of detection. Here, we describe a measure of robustness for normalized correlation, derived in Cox *et al.* [95] and tested in Miller *et al.* [290]. This measure estimates the amount of white Gaussian noise that can be added to the embedded vector, $\mathbf{v_w}$, before it is expected to fall outside the detection region. The reason for basing this measure on Gaussian noise, rather than on some more realistic form of degradation, is that correlation coefficients are not inherently robust against it. If we ensure that our watermarks are robust against this type of

noise, we help to ensure they are robust against other types of degradations, for which the correlation coefficient may be better suited.

The estimated amount of noise the mark can survive is derived as follows. If we add a noise vector, $\mathbf{n}$, to $\mathbf{v_w}$, the resulting normalized correlation is given by

$$z_{\text{nc}}(\mathbf{v_w} + \mathbf{n}) = \frac{(\mathbf{v_w} + \mathbf{n}) \cdot \mathbf{w_r}}{|\mathbf{v_w} + \mathbf{n}||\mathbf{w_r}|}. \tag{5.11}$$

Assuming the noise vector is likely to be orthogonal to both $\mathbf{v_w}$ and $\mathbf{w_r}$, we have

$$z_{\text{nc}}(\mathbf{v_w} + \mathbf{n}) \approx \frac{\mathbf{v_w} \cdot \mathbf{w_r}}{\sqrt{\mathbf{v_w} \cdot \mathbf{v_w} + \mathbf{n} \cdot \mathbf{n}}|\mathbf{w_r}|}. \tag{5.12}$$

We are interested in finding the magnitude of noise, $R = \sqrt{\mathbf{n} \cdot \mathbf{n}}$, that causes $z_{\text{nc}}(\mathbf{v_w} + \mathbf{n})$ to fall below the threshold, τ_{nc}. The embedder will work with any value that is a monotonic function of R. Therefore, we will be satisfied, instead, with the maximum allowable value of R^2. Replacing $z_{\text{nc}}(\mathbf{v_w} + \mathbf{n})$ with τ_{nc} in Expression 5.12, and solving for R^2, leads to

$$R^2 = \left(\frac{\mathbf{v_w} \cdot \mathbf{w_r}}{\tau_{\text{nc}}|\mathbf{w_r}|}\right)^2 - \mathbf{v_w} \cdot \mathbf{v_w}. \tag{5.13}$$

This gives us a rough measure of how robustly $\mathbf{v_w}$ represents $\mathbf{w_r}$.

We now wish to make an embedder that holds R^2 constant, rather than holding z_{nc} constant. Contours of constant R^2, for a given threshold, are shown in Figure 5.7. As can be confirmed algebraically, these contours are hyperbolae. Note that each hyperbola has a positive and a negative part. We wish to ensure that the correlation between the watermark and the selected vector, $\mathbf{v_w}$, is positive. Therefore, we restrict our embedder to only the positive portion of the hyperbola. In other words, the embedder will not only hold R^2 constant, but also ensure that z_{nc} is positive. Thus, our embedding region in N-dimensional marking space is not a cone but one-half of an N-dimensional, two-sheet hyperboloid.[1]

The embedder must find the point, $\mathbf{v_w}$, on the hyperboloid that is closest to a given point, $\mathbf{v_o}$. Although this problem can be solved analytically, the solution involves solving a quartic equation, which is complicated to implement. For the sake of simplicity, therefore, we implement our next example system using an exhaustive search.

[1] A hyperboloid is obtained by rotating a hyperbola about one of its axes. With hyperbolae oriented like those in Figure 5.7, a rotation about the *y*-axis will result in a single surface, called a *one-sheet* hyperboloid. A rotation about the *x*-axis will result in two surfaces (one for the positive part and one for the negative part), or a *two-sheet* hyperboloid. We restrict the embedding region to the positive sheet of the two-sheet hyperboloid.

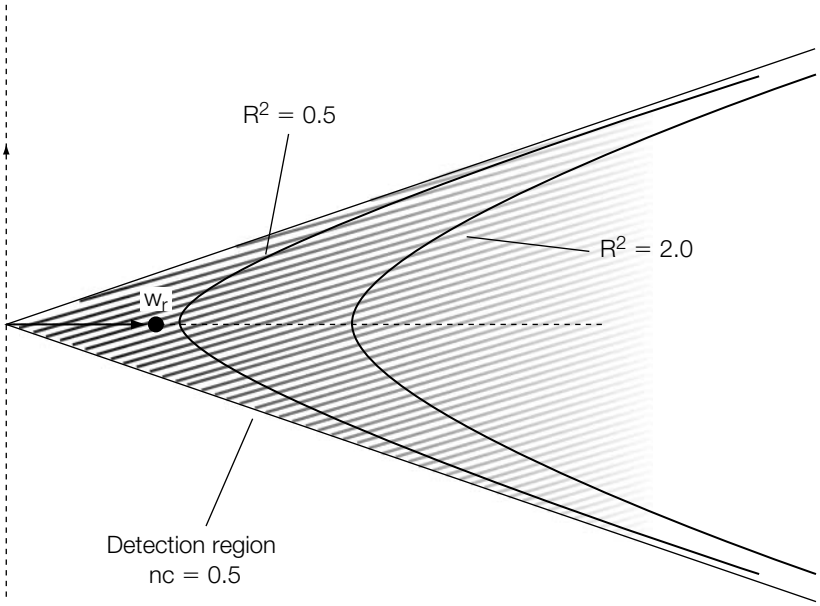


FIGURE 5.7

Normalized correlation detection region with two constant robustness embedding contours.

INVESTIGATION

Fixed Robustness Embedding for a Normalized Correlation Detector

We now present a watermarking system that embeds according to the robustness estimate given in Equation 5.13.

System 8: E_BLK_FIXED_R/D_BLK_CC

The E_BLK_FIXED_R embedder is essentially the same as the E_BLK_FIXED_CC embedder, except that it maintains a constant R^2 instead of a constant normalized correlation. The target value of R^2 is provided by the user as a "strength" parameter. This is instead of the β value used in E_BLK_FIXED_CC.

As mentioned previously, rather than finding $\mathbf{v_w}$ analytically, we perform a simple, exhaustive search, illustrated in Figure 5.8. This figure lies in the same **XY** plane as used for fixed normalized correlation embedding. We test several possible values of $y_{\mathbf{v_w}}$ within the range of 0 to $y_{\mathbf{v_o}}$, computing for each the value of $x_{\mathbf{v_w}}$ that results in a point on the hyperbola. This can be found analytically by

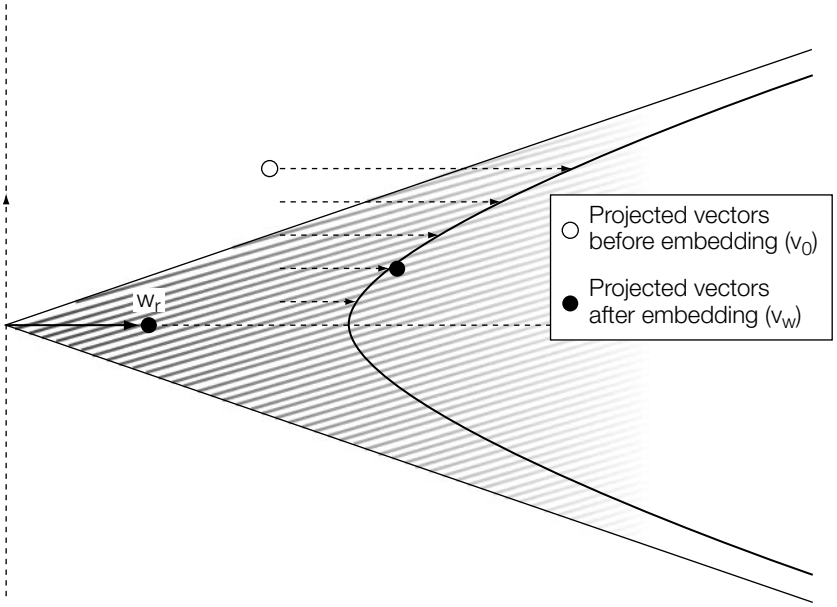


FIGURE 5.8

Illustration of exhaustive search for the closest point on a hyperbola. Each dashed line points to one search location. The search location resulting in the shortest distance to $\mathbf{v_o}$ is chosen.

substituting the vectors $x_{\mathbf{v_w}}, y_{\mathbf{v_w}}$ and 1,0 into Equation 5.13 for $\mathbf{v}$ and $\mathbf{w_r}$, respectively, and then solving for $x_{\mathbf{v_w}}$, which yields

$$x_{\mathbf{v_w}} = \sqrt{\frac{\tau_{nc}^2\left(R^2 + y_{\mathbf{v_w}}^2\right)}{1 - \tau_{nc}^2}}. \tag{5.14}$$

For each resulting point, $x_{\mathbf{v_w}}, y_{\mathbf{v_w}}$, we compute the distance to $x_{\mathbf{v_o}}, y_{\mathbf{v_o}}$. The point with the smallest distance yields the final values of $x_{\mathbf{v_w}}, y_{\mathbf{v_w}}$ used in Equation 5.9 to obtain $\mathbf{v_w}$.

Experiments

Figures 5.9(a) and (b) show the performance of the E_BLK_FIXED_R embedding algorithm for a given detection threshold, τ_{nc}, and two different values of R^2; namely, $R^2 = 10$ and $R^2 = 30$. Note that the range of detection values obtained after embedding watermarks is wider than that obtained from E_BLK_FIXED_CC. This is because the E_BLK_FIXED_R embedder only ensures that the detection value is above the threshold, τ_{nc}. Beyond that, it is willing to trade the magnitude of the detection value for higher robustness.

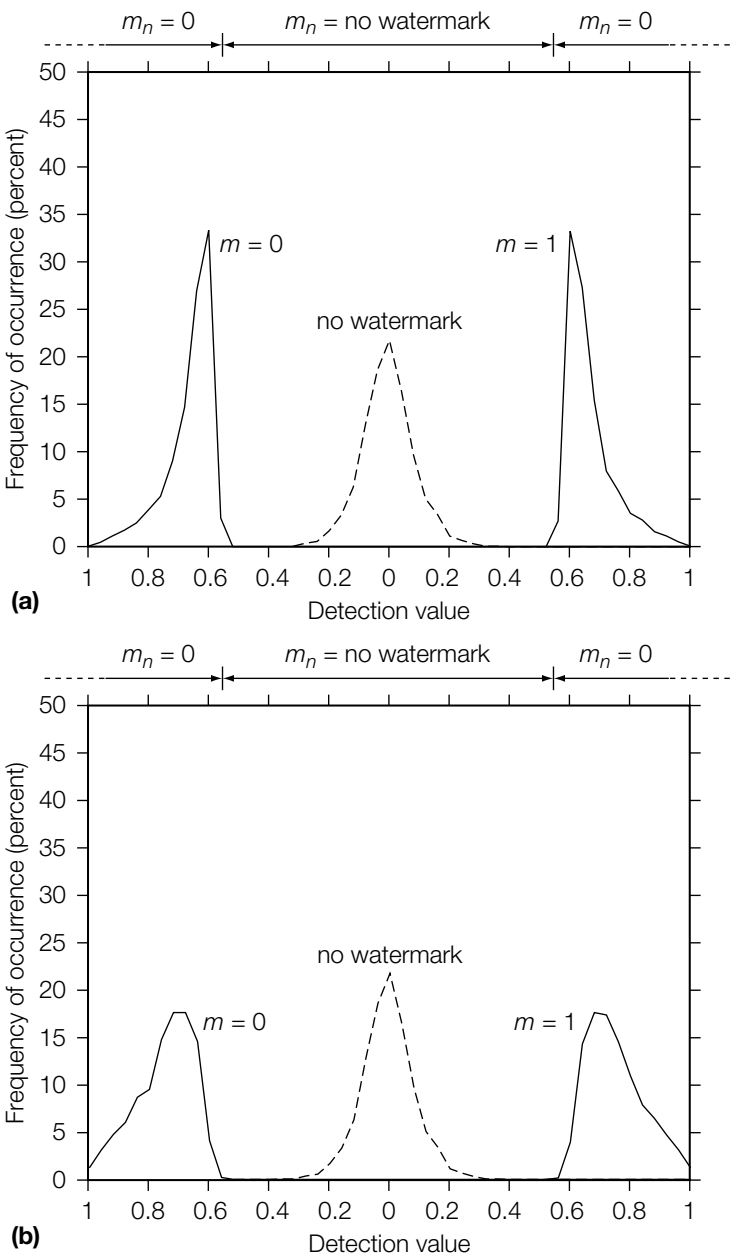


FIGURE 5.9

Distribution of detection values for two different values of R^2 in the E_BLK_FIXED_R/D_BLK_CC system. The strength parameter for these experiments was (a) $R^2 = 10$ and (b) $R^2 = 30$.

Figure 5.10 shows three histograms of the R^2 robustness values. The left, solid line is the result of using the E_BLK_FIXED_R embedder with a target R^2 value of 10, and the right solid line is the result of a target R^2 value of 30. The dotted line is the result of using the E_BLK_FIXED_CC embedder with $\beta = 0.40$. Note that the E_BLK_FIXED_CC embedder yielded many images with very low R^2.

To verify that increasing R^2 does indeed increase the robustness of the watermark, we performed the same robustness experiment on the E_BLK_FIXED_R images as performed on the E_BLK_FIXED_CC images. For a target robustness value of $R^2 = 10$, we found that about 80% of the watermarks were correctly detected. When the target robustness was increased to $R^2 = 30$, the percentage of correctly detected watermarks increased to almost 99%. This shows that the watermarks embedded using a higher R^2 target value were significantly more likely to survive the addition of noise.

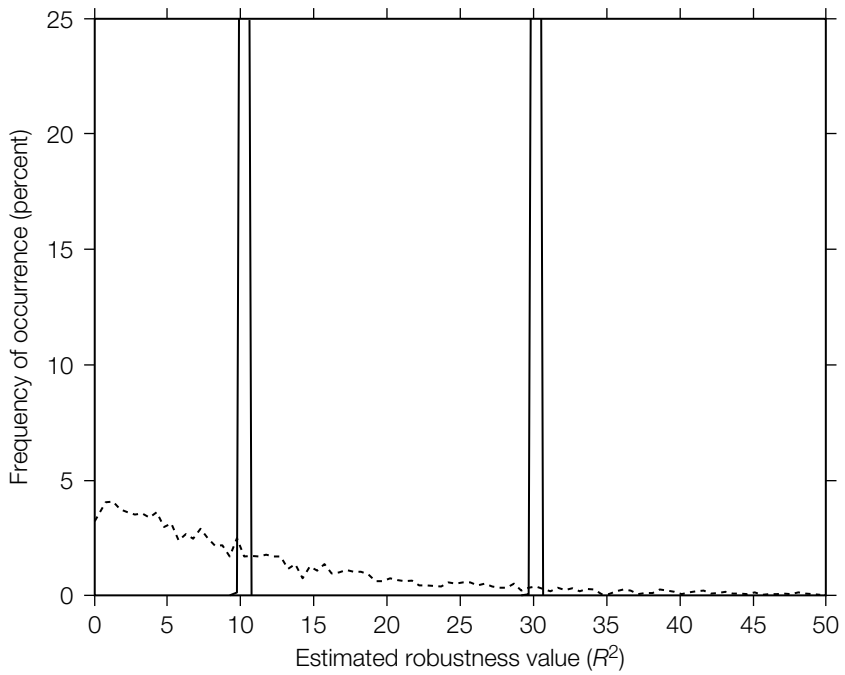


FIGURE 5.10

Comparison of robustness values for fixed correlation coefficient embedding (dashed line) and fixed robustness embedding (solid lines). The tops of the two histograms for the fixed robustness of $R^2 = 10$ and $R^2 = 30$ have been clipped so that more of the detail of the fixed correlation coefficient embedding robustness can be seen.

5.2 WATERMARKING USING SIDE INFORMATION

In this section we look at watermark embedding and detection from a more abstract point of view. In particular, we view watermarking as a communication problem, where the goal of the sender is to send a message m to a receiver. The sender is given an original Work, $\mathbf{c_o}$, and is allowed to modify it within a certain limit. The resulting watermarked Work, $\mathbf{c_w}[m]$, is sent to a receiver for decoding. However, the watermarked Work, $\mathbf{c_w}[m]$, may be degraded by a channel that is allowed to modify the watermarked Work, $\mathbf{c_w}[m]$, within a certain (different) limit. From the degraded Work, $\mathbf{c_{wn}}[m]$, the receiver will compute an estimate, $\hat{m}$, of the original message, m. In this chapter we study the question of the optimal strategy for the sender and the receiver, in the context of a degrading channel, such that the probability of error $m \neq \hat{m}$ is negligibly small. Note that an optimal strategy does not necessarily compute some intermediate marking signal, $\mathbf{w}[m]$, which is then added to $\mathbf{c_o}$ to produce $\mathbf{c_w}[m]$.

5.2.1 Formal Definition of the Problem

An abstract view of watermarking with side information is depicted in Figure 5.11. A sender wishes to send a message, m, to the receiver. To do so, the sender finds a watermarked Work, $\mathbf{c_w}[m]$, given (1) an original cover Work, $\mathbf{c_o}$, (2) a message, m, and (3) auxiliary information, K_e. The original Work, $\mathbf{c_o}$, is drawn from some distribution, $\mathcal{W}$, determined by the context of the application. For example, if the application is watermarking of family pictures, there will be a high likelihood of pictures with children and a low probability of pictures with mathematical equations. We assume, without loss of generality, that the messages are drawn from a uniform distribution (i.e., all messages are equally likely). The auxiliary information is additional information available to the encoder that assists in the watermarking process. Some of this auxiliary

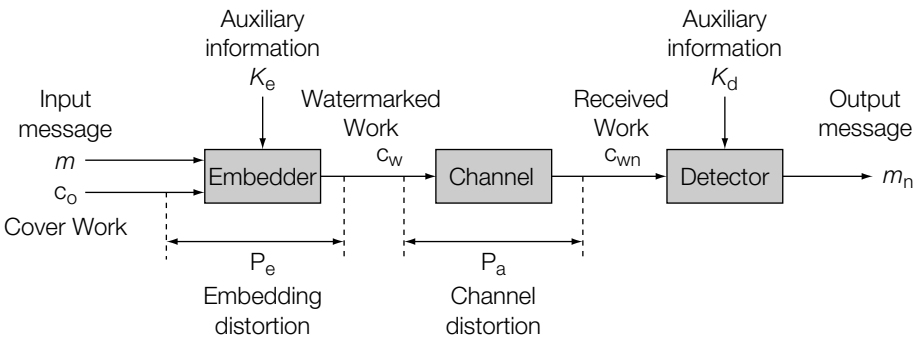


FIGURE 5.11

An abstract view of watermarking.

information may be shared with the decoder (e.g., the watermarking key in the examples of the previous chapters), while other information may be only available to the encoder.

The embedding process is constrained to produce a watermarked Work that is within a specified distortion limit (fidelity constraint) from the original. Thus, given a distortion metric, d, the distortion $d(\mathbf{c_o}, \mathbf{c_w}[m])$ between the original and watermarked Work must be less than a given threshold, P_e.

After watermark embedding, but before watermark detection, the watermarked Work may undergo further distortion (i.e., the "transmission" from the sender to the receiver may be degraded by an interfering channel, C). The channel, C, degrades the watermarked Work, $\mathbf{c_w}[m]$, to a received Work, $\mathbf{c_{wn}}[m]$. Since the channel models all distortions between the time of embedding and the time of detection, it includes both normal processing (e.g., lossy compression) as well as malicious attacks intended to remove the watermark. However, for the Work to have value to the receiver, the received Work must still resemble the watermarked Work. This condition is modeled as a constraint on the channel, C, such that $d(\mathbf{c_w}[m], \mathbf{c_{wn}}[m]) < P_a$. The threshold P_a may be much larger than P_e, reflecting, for example, the fact that consumers purchasing pirated content may have accepted a significantly degraded copy, while content owners seek to maintain very high fidelity. Degradations larger than P_a are considered to produce Works of unacceptable quality. Finally, using the auxiliary information, K_d, and the received Work, $\mathbf{c_{wn}}[m]$, the decoder produces an estimate, $\hat{m}$, of the original message, m. As indicated earlier, part of this auxiliary information may be shared with the encoder.

The performance of the watermarking scheme is measured by, among other things, the probability that the decoder correctly estimates the transmitted message. More commonly, this performance is measured by the transmission error rate, defined as the probability that the input message, m, and the decoded message, $\hat{m}$, are different. Another performance measure is the rate, R, of the scheme, defined as $\log_2(|\mathcal{M}|)/n$, where n is the length of the signals involved. More simply, the rate of a watermarking system is the number of transmitted message bits per symbol in the cover Work.

Note that there is a slight subtlety in the definition of the rate of a watermarking system that relates to the notions of *media space* and *marking space*. The abstract view presented in this chapter considers the encoding and decoding processes as black boxes that embed and decode messages. The abstract model is not aware of the distinction between media space and marking space. If a watermarking scheme takes Works in media space as input, the rate of such a system is with respect to the size of the original cover Works, irrespective of the internal processing in the encoder and the decoder. In particular, an internal representation of a marking space is not relevant for determining the rate of a system. However, it is certainly possible to view the transformation to and from marking space as a preprocessing and postprocessing step, respectively. In such a case, the rate, R, for watermarking is defined with respect to

signals in marking space. Converting this marking space rate to media space rate is then determined by the details of the transformation from media space to marking space. In this chapter, unless explicitly stated otherwise, we will consider the inputs to a watermarking scheme to be the elements of media space.

For the remainder of this section we will assume that the statistical behavior of the channel, C, is fixed and known to the encoder and decoder. Under this assumption we say that the tuple $(R, P_e, P_a,)$ is *achievable* if, for every $\epsilon > 0$, there is a *combined embedding and detection scheme* with transmission rate larger than $R - \epsilon$, embedding distortion less than P_e, channel distortion less than P_a, and error rate less than ϵ.[2]

In its full generality we do not know which triples, (R, P_e, P_a), are achievable and what the associated optimal coding schemes are. However, in the case of Gaussian signals with MSE distortion and additive white Gaussian noise (AWGN) channels, we are able to characterize the achievable triples and the corresponding optimal watermark embedding and detection schemes. This will be the topic of the next sections.

5.2.2 Signal and Channel Models

Let S, Z, and N be zero mean white Gaussian random variables with power Q, P_e, and P_a, respectively, such that the triple (S, Z, N) is jointly Gaussian. These random variables will model the original Work, $\mathbf{c_o}$, the watermark, $(\mathbf{c_w}[m] - \mathbf{c_o})$, and the degradation due to the channel, C, respectively.

We model an original Work, $\mathbf{c_o}$, as an i.i.d. sequence $\mathbf{s} = \{s_1, s_2, \ldots, s_n\}$ over $\mathbf{S}$. More precisely, we model an original cover Work as a random sequence of real values, where each component is identically and independently drawn from the (same) probability distribution governing $\mathbf{S}$. This model might seem extremely simplistic and not relevant in practice. However, in many cases it is possible to represent and approximate Works in media space with this simple model. For example, the AC coefficients of the DCT representation of an image typically have a first-order representation as an i.i.d. Gaussian sequence. Moreover, this simple Gaussian model will be shown to provide guidelines for watermarking cover Works of a more complicated character.

Given a message, m, uniformly drawn from a message set, $\mathcal{M}$, the sender produces a watermarked Work, $\mathbf{x} = \{x_0, x_1, \ldots, x_n\}$, such that distortion $d(\mathbf{s}, \mathbf{x}) < P_e$, where the distortion metric, $d(\mathbf{s}, \mathbf{x})$, is the MSE defined as

[2] More precisely, the actual requirement is that for every $\delta > 0$ there is a *combined embedding and detection scheme* with rate larger than $R - \delta$, embedding distortion less than $P_e + \delta$, attacker distortion less than $P_a + \delta$, and error rate less than $\epsilon < \delta$. We will not pursue this level of mathematical detail in this book. The interested reader is referred to Moulin and Sullivan [303].

$$d(\mathbf{s}, \mathbf{x}) = \frac{1}{n} \sum_i (s_i - x_i)^2.$$

At this point we do not specify *how* the encoder creates the watermarked Work, $\mathbf{x}$. However, intuitively, the larger the embedding power, P_e, the better able the transmitter is to communicate the message, m, albeit at the expense of a larger embedding distortion.

The channel, C, is modeled as an AWGN channel that modifies the watermarked work, $\mathbf{x}$, by adding to each sample, x_i, a value, ν_i, where $\mathbf{n} = \{\nu_1, \ldots, \nu_n\}$ is an i.i.d. sequence over N. The degraded signal $\mathbf{y} = \mathbf{x} + \mathbf{n}$ is given to the decoder to estimate the original message, m. This channel model might also seem too simple and not relevant in practice. However, this is only seemingly so, as in many practical cases signal degradations can be approximated by AWGN channels. For example, image degradations through quantization of DCT coefficients can be approximately modeled by an AWGN channel, provided the quantization step sizes are not too large. Intuitively, the larger the power, P_a, of the additive noise, $\mathbf{n}$, the more the channel, C, will prevent reliable communication between the transmitter and the receiver.

We will assume that the sender and the receiver are aware of the channel power, P_e, but are unaware of any specific noise value, ν_i. Moreover, the sender and receiver will share knowledge of the signal power, Q, the embedding power, P_e, and the message set, $\mathcal{M}$. The sender and receiver may also share other auxiliary information, but the receiver is not assumed to have any knowledge of the original work, $\mathbf{s}$.

With these assumptions we will consider progressively more complicated watermarking scenarios and state results on achievability regions as well as optimal watermarking methods. We start with the simplest case, in which the set of original cover Works consists of a single Work, $\mathbf{s}$.

5.2.3 Optimal Watermarking for a Single Cover Work

We consider the case of original cover Works with $Q = 0$. In this case there is only a single original cover Work, viz. the all-zero cover Work, $\mathbf{0}$.[3] Messages are transmitted as signals with power less than P_e and the AWGN channel modifies the watermarked signal, $\mathbf{x}$, by adding white Gaussian noise, $\mathbf{n}$, with power less than P_a.

This formulation shows that watermarking with a single cover Work is equivalent to the classical problem of communicating over an AWGN Gaussian

[3] We can slightly extend this case to the situation where both the sender and the receiver agree upon and have knowledge of a single, not necessarily all-zero, cover Work. By subtracting the fixed cover Work from the transmitted and received signals, this situation can be treated as if the cover Work is the all-zero signal.

channel: (1) given a message, m, the encoder chooses a signal, $\mathbf{x}$, within the power constraint, $P_{\mathbf{e}}$; (2) the decoder receives a signal, $\mathbf{y} = \mathbf{x} + \mathbf{n}$, where $\mathbf{n}$ is white Gaussian noise with power constraint, $P_{\mathbf{a}}$; and (3) the decoder estimates m from $\mathbf{y}$. The rate of such a system is given by the famous Shannon theorem on the capacity of an AWGN channel [365].

Theorem 1 (Shannon) *The rate of an additive white Gaussian noise (AWGN) channel is given by*

$$R = \frac{1}{2} \log \left(1 + \frac{P_{\mathrm{e}}}{P_{\mathrm{a}}} \right), \tag{5.15}$$

where P_{e} is the signaling power and P_{a} is the distortion power.

This theorem says that for the given power constraints, and for signals that are sufficiently long, the number of bits that can be transmitted per sample is equal to the right-hand side of Equation 5.15.

Shannon's result can actually be understood using a simple geometrical interpretation, which will be the topic of Section 5.2.5. This geometrical understanding will provide an intuitive understanding of more advanced results on watermarking. In particular, it will provide an insight into Costa's theorem that says that the Shannon rate can be achieved independently of the power, Q, of the original cover Work.

5.2.4 Optimal Coding for Multiple Cover Works

The previous section considered watermarking of a single cover Work for an AWGN channel. In this case, we showed that the rate of the system is given by Shannon's theorem on the capacity of an AWGN channel. In the more general case of Gaussian sources with nonzero power, the solution is more difficult and new tools are required.

In this more general case, the encoder has to transmit for each host signal, $\mathbf{s}$, and each message, m, a signal, $\mathbf{s}_{\mathrm{w}}$, that is within the embedding power constraint. At the same time the decoder needs to be able to retrieve the message, m, from the received signal, which has been degraded by AWGN noise present in the channel. We observe two opposing forces: On one hand, the code words for any message, m, need to be sufficiently dense to be able to embed within the embedding power constraint, and on the other hand, the set of code words for any two distinct messages need to be sufficiently sparse to allow an unambiguous and correct decision by the decoder.

As it turns out, an optimal set of code words can no longer be constructed by random sampling of a sum of two uncorrelated Gaussian random variables, which suffices to understand the Shannon AWGN theorem. Rather, the random variables involved will have a nonzero correlation, and we need a few tricks to compute the rate of information that can be transmitted over correlated Gaussian signals.

In the following sections we will sketch a geometrical interpretation of Gaussian signals. This will suffice to sketch a proof of the Shannon AWGN theorem. Next, we will consider transmitting information over correlated signals and introduce the notion of *mutual information* for Gaussian signals. Putting everything together, we are able to state and sketch a proof of Costa's theorem on dirty-paper coding.

5.2.5 A Geometrical Interpretation of White Gaussian Signals

Let S be a Gaussian random variable with variance σ^2. A geometric interpretation of i.i.d. signals over S states that for sufficiently large n, we may view the set of i.i.d. signals over S of length n as uniformly distributed within the n-dimensional ball, $B_n(\sigma)$, of radius σ in $\mathcal{R}^n$. This interpretation allows us to exploit the properties of balls in high dimensions.

- For large n, the volume of multidimensional balls is concentrated near the surface of the ball. This can be seen by considering the expression for the volume, V, of a ball of radius r in n dimensions:

$$V = K_n r^n, \tag{5.16}$$

  where K_n is a constant that only depends on the dimensionality, n. The ratio of the volume of a ball of radius r_1 to a ball of radius r_2, $r_1 < r_2$, is given by

$$V_1/V_2 = (r_1/r_2)^n, \tag{5.17}$$

  which can be made arbitrarily small if n is sufficiently large. It therefore follows that most of the volume of a high-dimensional ball is close to the surface (bounding sphere) of the ball.

- Using the uniform distribution assumption and the previous bullet, it follows that for large n, a randomly drawn signal, $\mathbf{s}$, will, with high probability, have MSE power close to σ^2 (i.e., the point will be close to the surface of the ball).

- For n sufficiently large, i.i.d. signals over S can be viewed as vectors of magnitude σ, and therefore as points on an n-dimensional sphere of normalized radius σ. Moreover, the volume of the ball bounded by this sphere is concentrated at this bounding sphere.

- Two independently drawn signals, $\mathbf{s}_1$ and $\mathbf{s}_2$, over S, have an expected normalized inner product equal to 0. Therefore, for n sufficiently large, we may represent two such signals by two orthogonal vectors. This representation of high-dimensional signals is sketched in Figure 5.12 where the signal $\mathbf{s}_2$ is shown as a vector whose origin is displaced to the endpoint of the vector representing $\mathbf{s}_1$.

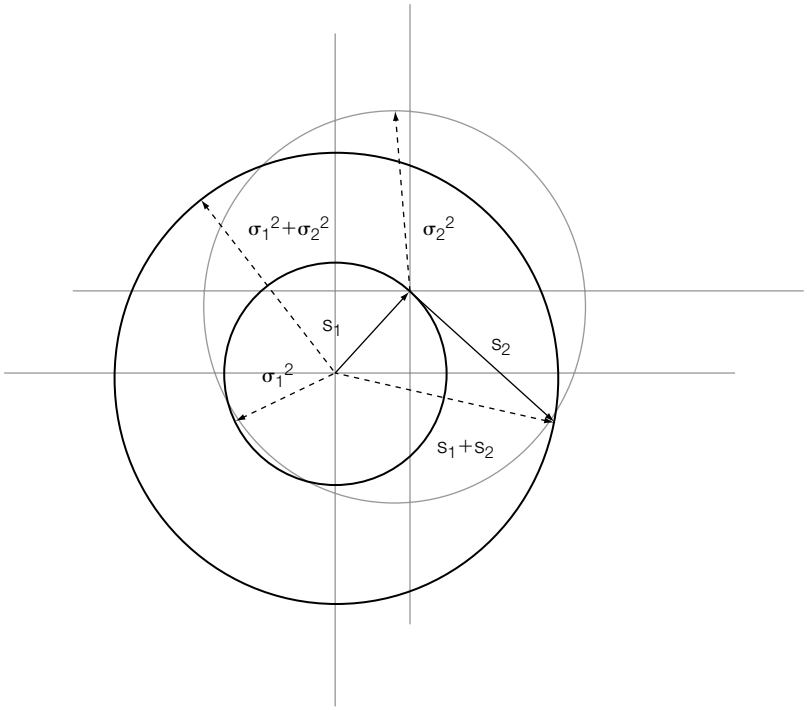


FIGURE 5.12

Signals represented as vectors.

In the next section we will see how this geometrical interpretation leads to a geometrical understanding of Shannon's theorem.

5.2.6 Understanding Shannon's Theorem

The typical information-theoretic construction for a sender and receiver to achieve optimal channel coding is to agree upon a common code book, $\mathcal{U} = \{\mathbf{u}_m\}_{m \in \mathcal{M}}$, where each element of $\mathcal{U}$ is a signal of normalized power, P_e, over a random variable, X (as always, the length of the signals is assumed to be sufficiently large). To send a message, m, the sender selects the signal $\mathbf{x} = \mathbf{u}_m$ and transmits it. The AWGN channel degrades the transmitted signal to a signal $\mathbf{y} = \mathbf{x} + \mathbf{n}$, where $\mathbf{n}$ is independent of $\mathbf{x}$ and of power P_a. In order to retrieve the message, m, the receiver searches the code book, $\mathcal{U}$, and finds the code word that best matches the received signal, $\mathbf{y}$. The index of that code word is taken as the estimate of the original message. If no distortions occur in the transmission process, the receiver can always retrieve the original message. However, when the received signal is distorted by an additive random signal, the receiver has a potential uncertainty on the original signal. This uncertainty can only be

resolved if the regions (balls) of uncertainty do not overlap. A simple counting argument then can be used to bound the number of messages that can reliably be exchanged as illustrated in Figure 5.13 and discussed below:

- The uncertainty introduced by the noise is represented by an "uncertainty" ball of volume, $K_n P_a^{n/2}$, where K_n is a dimension constant, as in Equation 5.16, and P_a is the noise power.
- The received signals must lie inside a ball of radius, $\sqrt{P_e + P_a}$, where P_e is the signal power.
- For error-free transmission, we require that the disks of uncertainty do not overlap. This implies that the total number of code words, $|\mathcal{U}|$, in the code book, $\mathcal{U}$, satisfies

$$|\mathcal{U}| \leq \frac{K_n\left(P_e + P_a\right)^{n/2}}{K_n\left(P_a\right)^{n/2}} = \frac{\left(P_e + P_a\right)^{n/2}}{\left(P_a\right)^{n/2}} = \left(1 + \frac{P_e}{P_a}\right)^{n/2}. \tag{5.18}$$

- It then follows that rate, $R = \log\left(|\mathcal{U}|\right)/n$, satisfies

$$R \leq \frac{1}{2}\log\left(1 + \frac{P_e}{P_a}\right). \tag{5.19}$$

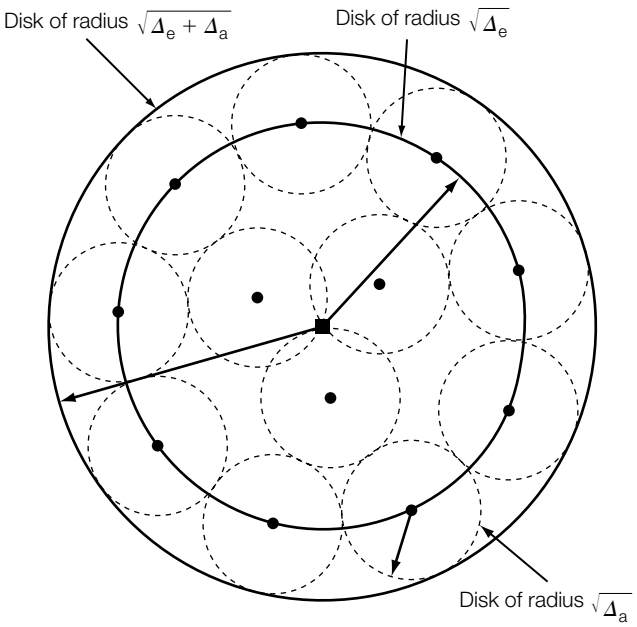


FIGURE 5.13

A geometrical interpretation of Shannon's AWGN theorem.

- It can be shown that, for n sufficiently large, and for a random code book, the relative volume of the gaps between the balls of uncertainty is as small as desired. Therefore, we can construct channel codes with R as close to $\frac{1}{2}\left(1+\frac{P_c}{P_a}\right)$ as desired.
- It follows that the maximal rate for an AWGN channel is given by the Shannon formula as stated in Theorem 1.

In summary, for random variables X, N, and Y, with X and N independent and $Y = X + N$ the maximal rate of transmission for signals over X and Y, is given as

$$\frac{1}{2}\log\left(\frac{\sigma_Y^2}{\sigma_N^2}\right). \tag{5.20}$$

In the next section, we study the transmission rate for signals over X and Y when Y is not necessarily of the form $X + N$. We will prove that the maximal rate of transmission can be expressed by the quantities $\mathrm{E}[X^2]$, $\mathrm{E}[Y^2]$, and $\mathrm{E}[XY]$, and coincides with the Shannon result when Y is of the form $X + N$. The results of this next section are essential for understanding the Costa result on optimal watermarking.

5.2.7 Correlated Gaussian Signals

Let X and Y be two jointly Gaussian random variables and let $\rho = E[XY]$ be the correlation between X and Y. In this section we consider the problem of communicating over the channel $X \rightarrow Y$ by transmitting signals over X and receiving signals over Y. An example of such a system is provided by the AWGN channel where $\mathbf{y}$ over Y is obtained as $\mathbf{x} + \mathbf{n}$, $\mathbf{n}$ over N, where X and N are independent. We will show that the more general case of correlated signals can be reduced to this AWGN case, allowing an expression for the maximal transmission rate. We refer to Figure 5.14 for a geometrical interpretation.

- In order to reduce to the AWGN case, we need to write Y as a sum of two independent Gaussian variables, where one of these variables is tightly coupled with the variable X. An obvious choice would be to write $Y = Z + \lambda X$ such that Z is independent from λX.
- As X and Y are jointly normal, to verify that $Z = Y - \lambda X$ is independent of λX, it is sufficient to verify that Z and λX are uncorrelated.[4] The geometrical interpretation of this condition is the vectors representing X and Z being orthogonal, as shown in Figure 5.14.
- The variable Z is uncorrelated with λX if, and only if,

$$\mathrm{E}[XY] = \lambda \mathrm{E}[X^2]. \tag{5.21}$$

[4] In general, independence implies uncorrelated, but not the other way around. For jointly Gaussian random variables the two notions are equivalent [320].

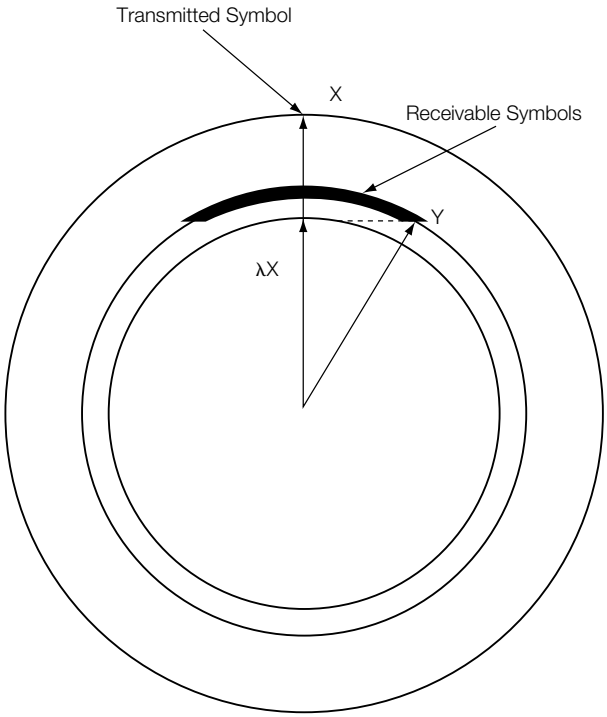


FIGURE 5.14

Coding for correlated signals.

- Recalling the proof of Shannon's theorem, the optimal coding strategy uses a code book, $\mathcal{U} = \{\mathbf{x}_m\}$, over the distribution, $\lambda\mathcal{X}$. Given a message, m, the sender selects the code word, $\mathbf{x}_m$, multiplies this by $1/\lambda$ to restore the original power condition, and transmits it.
- From the receiver's perspective, the received signal, $\mathbf{y}$, is a noisy version of the code word, $\mathbf{u}_m$, degraded by additive noise, $\mathbf{z}$, over the random variable Z. The receiver may now use the same decoding strategy as discussed earlier for Shannon's original theorem, and estimate the message as the index of the best matching code word.
- The volume of the balls of uncertainty is determined by the power of the variable Z. A simple calculation yields:

$$\mathrm{E}[Z^2] = \mathrm{E}[Y^2] - \lambda^2\mathrm{E}[X^2] = \frac{\mathrm{E}[X^2]\mathrm{E}[Y^2] - \mathrm{E}[XY]^2}{\mathrm{E}[X^2]}. \tag{5.22}$$

- Recalling the proof of the original Shannon theorem, the rate of communication is determined by the ratio of the power of the received signals, Y, divided by the power of the additive noise component, Z:

$$\frac{\mathrm{E}[Y^2]}{\mathrm{E}[Z^2]}. \tag{5.23}$$

To summarize, we can formulate a generalization of Shannon's AWGN theorem stating the maximal rate of communication for the channel $X \rightarrow Y$ in terms of the power and correlation of the random variables X and Y.

Theorem 2 (Shannon) *Let X and Y be two jointly Gaussian random variables.*

- *The maximal rate R of communication for the channel $X \rightarrow Y$ is given by the expression $I(X;Y)$ defined as*

$$I(X;Y) = \frac{1}{2}\log\left(\frac{\mathrm{E}[X^2]\mathrm{E}[Y^2]}{\mathrm{E}[X^2]\mathrm{E}[Y^2] - \mathrm{E}[XY]^2}\right). \tag{5.24}$$

- *The maximal rate can be approached by constructing a random code book, $\mathcal{U} = \{\mathbf{x}_m\}$, over the distribution λX, where $\lambda = E[XY] / E[X^2]$. Sending a message, m, involves selecting the code word, $\mathbf{x}_m$, and transmitting $\mathbf{x}_m / \lambda$. Decoding involves finding the code word that best matches the received signal.*

- *Alternatively, the scaling by λ can be moved to the receiver. The maximal rate is achieved by constructing a codebook, $\mathcal{U} = \{\mathbf{x}_m\}$, over the random variable X. Sending a message, m, amounts to transmitting the code word, $\mathbf{x}_m$. Decoding involves finding the code word, $\mathbf{x}_{\hat{m}}$, that best matches the scaled version $\mathbf{y} / \lambda$ of the received signal.*

The expression $I(X;Y)$ is usually referred to as the *mutual information* between X and Y. The mutual information can be defined for arbitrary random variables and is a central concept in information theory [89]. For our purposes it is sufficient to restrict ourselves to the Gaussian case and the definition in Equation 5.24. Theorem 2 states that the maximal rate of information for two random variables X and Y is given by their mutual information $I(X;Y)$. This rate can be achieved to any degree of accuracy by choosing n sufficiently large and using random code books, $\mathcal{C}$. Sending a message, m, involves (1) the sender transmitting the signal, $\mathbf{x}_m$, and (2) the receiver finding the code word in $\mathcal{C}$ that is closest to a scaled version of the received signal. The scale factor, λ, depends on the statistics of the random variables X and Y and is given by Equation 5.21.

Note 1 *Viewing random variables as vectors and the expression $E[XY]$ as the inner product between two vectors, the mutual information $I(X;Y)$ can be interpreted as $-\log\left(|\sin(\beta)|\right)$, where β is the angle between the "vectors" X and Y.*

Note 2 *The best match for a received signal, $\mathbf{y}$, is the code word, $\mathbf{x}_m$, such that the angle between $\mathbf{y}$ and $\mathbf{x}_m$ is sufficiently close to α. As a consequence, the scaling operation at the sender or receiver can be avoided by choosing a random code book of the correct size over the random variable, X, and estimating the transmitted signal by computing and verifying the normalized inner product. This also implies, as angles are invariant to scalar*

multiplication, that the Shannon decoder is inherently robust to unknown global gains in the transmission channel.

5.3 DIRTY-PAPER CODES

In the following sections we will introduce the two main tools that are needed for optimal watermarking of Gaussian signals. The first and most important tool is a dirty-paper code. A dirty-paper code is an extension of the classical notion of a code as a set of code words, where each code word represents a single message. In contrast, in a dirty-paper code each message is represented by a *set* of code words (i.e., each message is represented by multiple code words). A dirty-paper code is therefore defined as a *collection* of codes. Transmission of a message, m, entails finding the code word that fits best the host signal into which m needs to be embedded.

The second tool is distortion compensation. Distortion compensation enables adaptation of a transmit symbol to the underlying host signal. Its distinguishing feature is that it adds fuzziness to quantization. In classical coding schemes, for a given message, m, a pair of host signals that are "close" are in general quantized to the same transmit signal (the same code word in the code book of all code words). This is no longer true when distortion compensation is applied.

In summary, dirty-paper codes and distortion compensation are two tools that serve the same purpose: adapting the transmit signal to the underlying host signal to achieve the maximal transmission rate at minimal distortion. The purpose of the following sections is to explain this in more detail by discussing geometric models for Gaussian signals, deriving basic results for the transmission rate over Gaussian signals, and stating Costa's main result on dirty-paper coding. The strategy that we will follow is to first relate watermarking of a collection of cover Works with one element to the classical Shannon theorem on transmission over AWGN channels. We then point out why the Shannon strategy will fail when our collection of cover Works is modeled as Gaussian signals with nonzero power. Introducing the tools of dirty-paper coding and distortion compensation we are able to state and prove Costa's surprising result that the impact of the host signal on the transmission rate can be avoided. We begin with a discussion on interpreting Gaussian signals.

5.3.1 Watermarking of Gaussian Signals: First Approach

Let S, X, and Y be random variables representing the original Work, the watermarked Work, and the degraded Work, respectively. Let $\mathcal{M}$ be a set of messages of size $|\mathcal{M}|$. The watermark embedding process is constrained by the maximal embedding distortion, $P_{\mathbf{e}}$. This implies that for every message, m, and every

signal, $\mathbf{s}$, there is at least one signal, $\mathbf{x}$, that both (1) represents the message, m, and (2) is close to the signal, $\mathbf{s}$. How can this be accomplished?

As a first solution, closely paralleling the original Shannon construction, select for each message, m, a code book, $\mathcal{C}_m$ (i.e., each message is now represented by more than one code word). Given the message, m, the watermarked Work, $\mathbf{x}$, is such that $\mathbf{x} \in \mathcal{C}_m$ and is sufficiently close to the original signal, $\mathbf{s}$ (i.e., the selected code word is as close as possible to the original signal, $\mathbf{s}$). Upon receiving a degraded signal, $\mathbf{y}$, the receiver finds the code word, $\mathbf{x}'$, in the union $\mathcal{C} = \cup_m \mathcal{C}_m$ of all subcode books, $\mathcal{C}_m$, and reports the index of the subcode book of which $\mathbf{x}'$ is a member.

We may think of the above approach as analogous to a high-dimensional quantizer where, for a for given message, m, all signals, $\mathbf{s}$, that are close to $\mathbf{x} \in \mathcal{C}_m$, are represented by $\mathbf{x}$. For example, consider a simple scalar equivalent. Let us assume a binary message set, $\mathcal{M} = \{0, 1\}$, where the two corresponding code books, $\mathcal{C}_0$ and $\mathcal{C}_1$, consist of the even integers and the odd integers, respectively. The sender sends a message 0 or 1 by transmitting the closest even integer or odd integer to the unwatermarked signal, $\mathbf{s}$. The receiver decides upon the message 0 or 1 by determining the oddness or evenness of the closest integer in the union of the two code books.

A code book, $\mathcal{C}$, that is a disjoint union of multiple subcode books, $\mathcal{C}_m$, is often referred to as a *dirty-paper code*. Each element $\mathbf{x} \in \mathcal{C}$ belongs to precisely one subcode book, $\mathcal{C}_m$, where m is referred to as the *syndrome*, syn($\mathbf{x}$), of the code word $\mathbf{x}$.

The term "dirty paper" derives from Costa's paper [88], entitled "Writing on Dirty Paper," in which he used the following analogy:

> Now imagine a sheet of paper covered with independent dirt spots of normally distributed intensity. In some sense, the problem of writing a message on this sheet of paper is analogous to that of sending information through the channel.... The writer knows the location and intensity of the dirt spots, but the reader cannot distinguish them from the ink marks applied by the writer.

When writing a symbol (say the letter "A") on dirty paper, it is best done by adapting that symbol to the dirt spots already present on the paper (informed writing, see Figure 5.15).

In the context of watermarking, the dirty paper is our cover Work, which is known to the watermark embedder (writer), but not to the detector (reader). By having multiple codes for each message, we can select the most appropriate code for the specific Work, and thereby remove the interference due to the original Work (dirty paper).

The quality of dirty-paper codes depends on two factors. The first factor is the density of the subcode words. The higher the density, the less the embedding distortion. This is because for a given message, the larger its associated

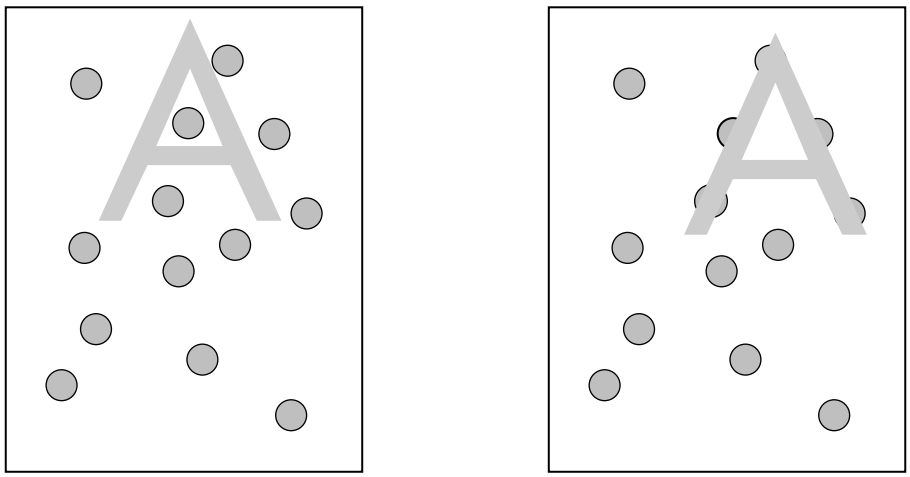


FIGURE 5.15

Writing on dirty paper.

subcode book is, the more likely we can find a code word that is close to the original signal, $\mathbf{s}$. The second factor is the density of the union code book. The lower the density, the better robustness we have. This is because the fewer code words present in the union code book, the more separated the code words can be, and thus the less likely we are to make a decoding error due to noise. Finding the balance between these two contradictory factors is at the heart of the design of dirty-paper codes.

INVESTIGATION

Informed Coding and Informed Embedding in a System Using Normalized Correlation Detection

This system combines informed embedding with a simple dirty-paper code. Here, we modify the E_BLK_FIXED_R/D_BLK_CC system (5.1.3) to represent each possible message with more than one reference mark. These reference marks play the role of the code words, $\mathbf{u} \in \mathcal{U}$, in the foregoing discussion.

We show experimentally that increasing the numbers of reference marks per message and choosing the reference mark that fits the cover Work best, leads to better performance. Specifically, watermarks with the same estimated robustness and false positive probabilities can be embedded with better fidelity.

System 8: E_DIRTY_PAPER/D_DIRTY_PAPER

This watermarking system is a simple extension of the E_BLK_FIXED_R/D_BLK_CC system. As in that system, watermark vectors are extracted by summing 8×8 blocks, and the presence of a watermark is tested with the correlation coefficient.

The system uses two sets of reference marks, $\mathcal{W}_0$ and $\mathcal{W}_1$, to encode one bit of information. Each mark in each of the two sets is generated by a Gaussian random number generator. The initial seed to the random number generator serves as the key.

When embedding a 0, the *dirty*-paper embedder first extracts a vector from the Work to be watermarked. It then finds the reference mark in $\mathcal{W}_0$ that has the highest correlation with this extracted vector. Finally, this mark is embedded using the same constant robustness algorithm employed in the E_BLK_FIXED_R embedder. The same process is used to embed a 1, except that the reference mark is drawn from $\mathcal{W}_1$.

The *dirty*-paper detector extracts a vector from its input image. It then computes the correlation coefficient between this vector and all reference marks in both $\mathcal{W}_0$ and $\mathcal{W}_1$. If the highest such correlation coefficient is below a given detection threshold, τ_{CC}, the detector reports that no watermark is present. Otherwise, if the best match is a mark in $\mathcal{W}_0$, the detector reports that a 0 was embedded, and if it is a mark in $\mathcal{W}_1$, the detector reports that a 1 was embedded.

The behavior of this system is illustrated in Figure 5.17i. The shaded area in this figure represents the detection region for one message. Because the message can be represented with any of several reference marks, its detection region is the union of several cones. The curve within each cone indicates a contour of constant robustness. The open circles represent unmarked Works. The arrows and filled circles indicate the behavior of the embedding algorithm, which finds the closest cone in the detection region and then moves the Work to the closest point of sufficient robustness within that cone.

Because the embedder and detector both use exhaustive search to find the best matches in $\mathcal{W}_0$ and $\mathcal{W}_1$, this system is only practical when the size of these sets is small.

Experiments

We tested the *dirty*-paper system with various numbers of reference marks in $\mathcal{W}_0$ and $\mathcal{W}_1$. In each test, we made two versions of each of 2,000 images: one with a 0 embedded and one with a 1 embedded.

The detection threshold was chosen to yield an overall false positive probability of roughly 10^{-6}. This probability was computed by first estimating the probability that a random Work will lie within one detection cone (using a method described

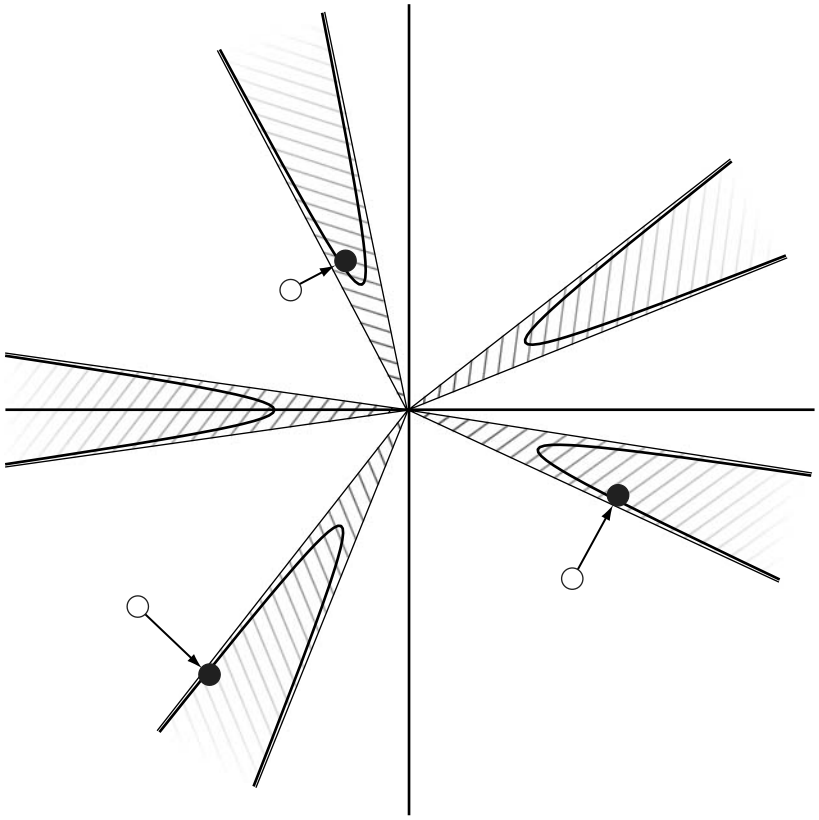


FIGURE 5.17i

in Chapter 7), and then multiplying that estimate by the total number of cones in the system; namely, $|\mathcal{W}_0| + |\mathcal{W}_1|$. This means that tests conducted with larger numbers of reference marks had to use higher detection thresholds (narrower cones) to maintain the same false positive probability.

In addition to constant false positive probability, we also had constant payload (always 1 bit) and constant estimated robustness (because we used the constant robustness embedding algorithm of E_BLK_FIXED_R). This means that the only performance parameter that should change as we vary the number of reference marks is fidelity.

Figure 5.18i shows the average mean squared error between original and watermarked versions of images, as a function of the number of reference marks. This indicates that as we increased from 1 to 40 reference marks per message the amount of distortion required to embed a watermark decreased by about 30%.

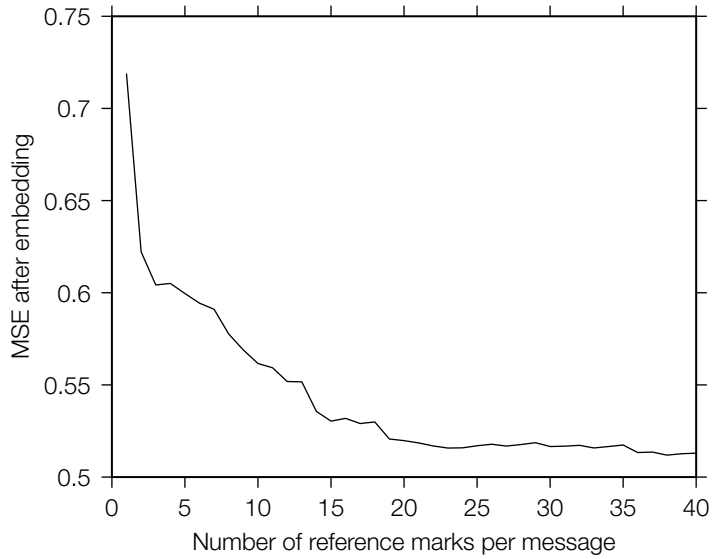


FIGURE 5.18i

We now present a construction of a dirty-paper code for the AWGN channel according to the principles of the previous section (i.e., by using random code books over random Gaussian variables). This is not an optimum code, but serves as a stepping-stone to the construction due to Costa. Specifically,

- Let Z be a Gaussian random variable, jointly normal and uncorrelated with S. Let $\mathrm{E}[Z^2] = P_{\mathbf{e}}$ and let $X = S + Z$.
- Let n be sufficiently large. For each message, m, a random code book, $\mathcal{C}_m$, over Z is chosen that has sufficiently many elements, such that the rate of the code book obeys

$$R(\mathcal{C}_m) \geq I(S;X) = \frac{1}{2}\log\left(\frac{\mathrm{E}[X^2]}{\mathrm{E}[Z^2]}\right). \tag{5.25}$$

- Following the proof of Shannon's theorem then, for every signal, $\mathbf{s}$, over S, there is a code word, $\mathbf{x} \in \mathcal{C}_m$, such that the distortion $d(\mathbf{s}, \mathbf{x})) \leq \mathrm{E}[Z^2] = P_{\mathbf{e}}$. In other words, the subcode books are sufficiently dense that the embedding constraint can be met.
- Moreover, for n sufficiently large, and choosing $R(\mathcal{C}_m) = I(S;X)$, the expected value of the distortion, $\mathrm{E}[d(\mathbf{s}, \mathbf{x})]$, is equal to $\mathrm{E}[Z^2]$.
- Finally, let $\mathcal{C}$ be the union of all code books (i.e., $\mathcal{C} = \cup_m \mathcal{C}_m$).

In order to communicate a message, m, for a given cover Work, $\mathbf{s}$, the sender transmits the code word, $\mathbf{x} \in \mathcal{C}_m \subset \mathcal{C}$, that is closest to the cover Work, $\mathbf{s}$. As argued above, such a code word can always be found with the expected distortion constraint. Now, let N be a Gaussian random variable with power $\mathrm{E}[N^2] = P_a$, representing the channel noise. Using a similar argument as in the proof of Shannon's theorem, the receiver will be able to reliably estimate the transmitted symbol if the density of the union code book is sufficiently low. More precisely, the rate $R(\mathcal{C})$ of the union of all the code books must satisfy

$$R(\mathcal{C}) \leq I(X;Y), \tag{5.26}$$

where the received signal, $Y = X + N$. Moreover, for n sufficiently large, a coding rate can be achieved that is as close to $I(X;Y)$ as desired.

Recalling that each message is represented by multiple code words (with rate $I(S;X)$), we derive the following theorem.

Theorem 3 *Using a random code book on the variable $X = S + Z$, the construction above yields a transmission rate, R_1, equal to*

$$R_1 = I(Y;X) - I(S;X). \tag{5.27}$$

Evaluating the right-hand side of Equation 5.27, we find

$$\begin{aligned} R_1 &= \frac{1}{2}\log\left(\frac{\mathrm{E}[S^2] + \mathrm{E}[Z^2] + \mathrm{E}[N^2]}{\mathrm{E}[N^2]}\frac{\mathrm{E}[Z^2]}{\mathrm{E}[S^2] + \mathrm{E}[Z^2]}\right) \\ &= \frac{1}{2}\log\left(\frac{P_s + P_e + P_a}{P_a}\frac{P_e}{P_s + P_e}\right). \end{aligned} \tag{5.28}$$

It is easy to verify that, as expected, R_1 is never larger than the Shannon rate $\frac{1}{2}\log\left(\frac{P_e+P_a}{P_a}\right)$ that is obtained by assuming the absence of the host (that is, $P_s = 0$). Also note that R_1 is functionally dependent on the power of the host signal, S, and that as P_s approaches infinity, the rate of the system decreases to 0.

Theorem 4 (Lower bound on the capacity of the AWGN watermarking channel) *The capacity of the AWGN watermarking channel with parameters (P_s, P_e, P_a) has a lower bound of R_1, where R_1 is given by Equation 5.28. This lower bound can be achieved using a random code book over the random variable $X = S + Z$, where Z is Gaussian with power P_e.*

5.3.2 Costa's Insight: Writing on Dirty Paper

Max Costa, in his landmark paper "Writing on Dirty Paper" [88], observed that that the actual capacity of the AWGN watermarking channel is larger than the lower bound of Theorem 4.

Costa's insight was to write $X = S + Z$ as $X = (\alpha S + Z) + (1 - \alpha)S$ and to note that appropriate quantization of the variable $U_\alpha = \alpha S + Z$ maintains the embedding distortion constraint, P_e, on S.

Given α, $0 < \alpha \leq 1$, and a quantization function, Q, such that $d(\mathbf{s}, Q(\mathbf{s})) \leq P_e$, Costa considered the partial quantization function, Q_α, defined by

$$Q_\alpha(\mathbf{s}) = Q(\alpha\mathbf{s}) + (1 - \alpha)\,\mathbf{s}. \tag{5.29}$$

With this definition it is not hard to verify that Q_α satisfies the same constraint as $Q = Q_1$:

$$\begin{aligned} d(\mathbf{s}, Q_\alpha(\mathbf{s})) &= \|Q(\alpha\mathbf{s}) + (1 - \alpha)\mathbf{s} - \mathbf{s}\| \\ &= \|Q(\alpha\mathbf{s}) - \alpha\mathbf{s}\| \\ &\leq P_e. \end{aligned} \tag{5.30}$$

In other words, a distortion constraint, P_e, is obtained by quantizing a fractional portion, $\alpha\mathbf{s}$, and restoring the remainder, $(1 - \alpha)\mathbf{s}$.

If $Q = Q_{\mathcal{C}}$ is a quantizer for a code book, $\mathcal{C}$, we can rewrite Q_α as follows:

$$\begin{aligned} Q_\alpha(\mathbf{s}) &= Q_{\mathcal{C}}(\alpha\mathbf{s}) + (1 - \alpha)\mathbf{s} \\ &= \alpha Q_{\alpha^{-1}\mathcal{C}}(\mathbf{s}) + (1 - \alpha)\mathbf{s} \\ &= \alpha Q_{\mathcal{D}}(\mathbf{s}) + (1 - \alpha)\mathbf{s} \\ &= Q_{\mathcal{D}}(\mathbf{s}) + (1 - \alpha)(\mathbf{x} - Q_{\mathcal{D}}(\mathbf{s})), \end{aligned} \tag{5.31}$$

where the code book, $\mathcal{D}$, is defined as $\alpha^{-1}\mathcal{C}$. The distortion, $P_{\mathcal{D}}$, of the quantizer, $Q_{\mathcal{D}}$, is obviously related to $P_{\mathcal{C}}$ as $P_{\mathcal{D}} = P_{\mathcal{C}}/\alpha^2$. The right-hand side of Equation 5.31 can therefore be interpreted as a quantization term $Q_{\mathcal{D}}(\mathbf{s})$ with power $P_{\mathcal{C}}/\alpha^2$ compensated by a term $(1 - \alpha)(\mathbf{x} - Q_{\mathcal{D}}(\mathbf{s}))$ that restores a factor $1 - \alpha$ of the quantization error. Another equivalent way of looking at this is provided by the following equation:

$$Q_\alpha(\mathbf{s}) - Q_{\mathcal{D}}(\mathbf{s}) = \alpha(Q_{\mathcal{D}}(\mathbf{s}) - \mathbf{s}), \tag{5.32}$$

which states that the additive offset induced by Q_α is a fraction, α, of the offset induced by quantizing to a scaled code book, $\mathcal{D}$.

The technique of partial quantization is known in the literature as *distortion compensated quantization* (DCQ) and α as the *distortion compensation parameter*. Figure 5.16 sketches DCQ for the case where the code book, $\mathcal{C}$, is a random code book over the variable $\alpha S + Z$, where Z is independent of S and of power, P_e. The figure shows that for $\alpha < 1$ every host signal, $\mathbf{s}$, is mapped to its own unique quantized version. In contrast to classical quantization, the output of a partial quantization function is not a discrete set of delta functions,

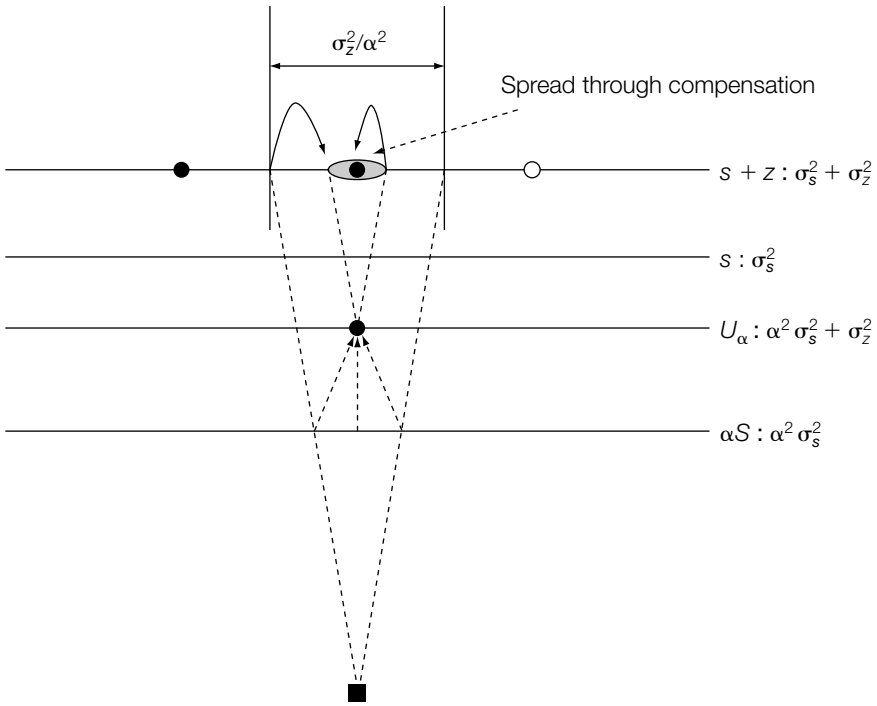


FIGURE 5.16

Partial quantization: a signal over $\mathcal{S}$ is scaled by α, quantized by a code book over $\alpha S+Z$ and restored to $S+Z$ by adding an original fraction $(1-\alpha)$. The expression $A : a$ denotes a random variable A and its power a.

but a set of functions with support centered around the restoration points of the quantizer, $Q_{\mathcal{D}}$, as illustrated in Figure 5.17.

To determine the capacity of Costa's method, the result of Theorem 4 can now be extended by substituting $\mathcal{C}_\alpha$ for $\mathcal{C}$ and constructing code books over $\mathcal{C}_\alpha$. The rate for this generalized watermarking method is denoted by R_α and, in complete analogy with Theorem 4, we have the following result.

Theorem 5 (Lower bound on the capacity of the AWGN watermarking channel (2)) *The capacity of the AWGN watermarking channel with parameters* (P_s, P_e, P_a) *is lower bounded by* R_α*, where* R_α *is given by*

$$R_\alpha = I(Y; U_\alpha) - I(S; U_\alpha), \tag{5.33}$$

where $U_\alpha = \alpha S + Z$, $0 < \alpha \leq 1$, *and* Z *is Gaussian with power,* P_e*, and independent of* S*. This lower bound can be achieved using a random code book over the random variable* U_α*.*

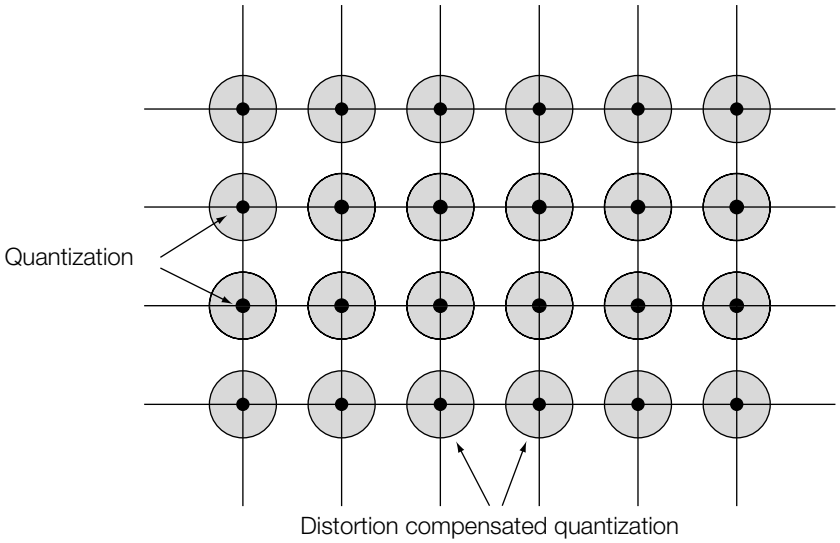


FIGURE 5.17

Quantization versus DCQ: a series of delta functions (black dots) versus regions with nonzero volume centered around quantization points (gray circles).

Note that the expressions $I(Y;U_\alpha)$ and $I(S;U_\alpha)$ correspond to the number of elements in the code book, $\mathcal{C}$, and subcode book, $\mathcal{C}_m$. Both terms are monotonically increasing in α. With respect to the term $I(S;U_\alpha) = I(\alpha S;\alpha S+Z)$, an increasing value of α implies a relatively smaller fixed-noise component, Z, and therefore a larger mutual information.

In geometric terms, an increasing value of α implies a larger volume in which the balls of uncertainty fit with respect to Z. Another equivalent geometrical interpretation is that as α increases, the angle between the two "vectors," S and U_α, decreases, implying an increase in mutual information.[5]

With respect to the term $I(Y;U_\alpha) = I(N+Z+S;\alpha^{-1}Z+S)$, an increasing value of α implies a decreasing systemic noise component, $\alpha^{-1}Z$, with respect to S. Geometrically, by increasing the value of α, the angle between the "vectors," $N+S+Z$ and $\alpha^{-1}Z+S$, is decreasing, implying an increase in mutual information (see Figure 5.18). There is no immediate intuitive qualitative result on the rate, R_α, as the difference between the two terms. However, using some simple algebra and Theorem 2, we can easily find explicit expressions for the mutual rate expressions and the rate R_α.

$$I(U_\alpha;S) = \frac{1}{2}\log\left(\frac{\alpha^2 P_s + P_e}{P_e}\right) \tag{5.34}$$

[5] Recall Note 2 at the end of Section 5.2.7 (p. 163); see Figure 5.18.

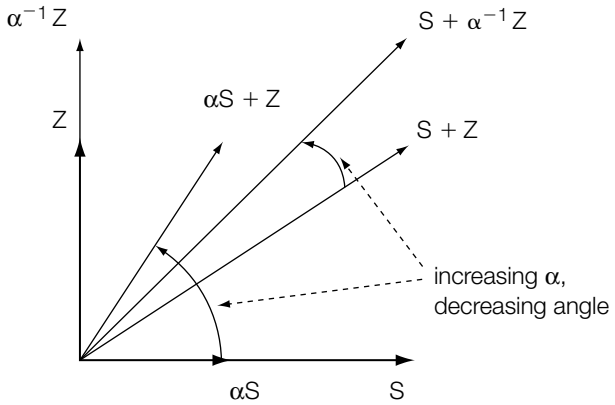


FIGURE 5.18

Dependency of mutual information terms on the distortion compensation parameter α.

$$I(U_\alpha; y) = \frac{1}{2}\log\left(\frac{(\alpha^2 P_s + P_e)(P_s + P_e + P_a)}{P_a(\alpha^2 P_s + P_e) + P_s P_e(1-\alpha)^2}\right) \tag{5.35}$$

$$R_\alpha = \frac{1}{2}\log\left(\frac{P_e(P_s + P_e + P_a)}{P_a(\alpha^2 P_s + P_e) + P_s P_e(1-\alpha)^2}\right). \tag{5.36}$$

Finding the maximal rate, R_α, is equivalent to finding the minimum of the expression $P_a(\alpha^2 P_s + P_e) + P_s P_e(1-\alpha)^2$. Differentiating with respect to α we find for the optimal α^*, that

$$\alpha^* P_a P_s - (1-\alpha^*) P_s P_e = 0. \tag{5.37}$$

We conclude that the optimal rate is achieved for $\alpha^* = P_e/(P_a + P_e)$, a value that is in general smaller than 1. Substituting α^* in Equation 5.36 we find

$$R_{\alpha^*} = \frac{1}{2}\log\left(1 + \frac{P_e}{P_a}\right). \tag{5.38}$$

Note that this rate is equal to the Shannon rate that can be achieved if both the sender and the receiver have knowledge of S. Therefore, this is the maximal rate that can be achieved for an AWGN watermarking system with parameters P_s, P_e, and P_a.

Theorem 6 (Costa) *The capacity of the AWGN watermarking channel $C(P_s, P_e, P_a)$ is given by*

$$C(P_s, P_e, P_a) = \frac{1}{2}\log\left(1 + \frac{P_e}{P_a}\right). \tag{5.39}$$

This rate can be achieved by setting the compensation parameter α equal to $\alpha^* = P_e/(P_a + P_e)$ and constructing a random code book over $U_{\alpha^*} = \alpha^* S + Z$, where Z is a Gaussian random variable of power, P_e.

This watermarking method is referred to as the *ideal Costa scheme* (ICS). Theorem 6 asserts that ICS achieves the maximal possible transmission rate for the AWGN watermarking channel. The core ingredients of ICS are dirty-paper codes in high-dimensional spaces and adaption to the statistics of the embedding and noise power through distortion compensation.

5.3.3 Scalar Watermarking

In the previous section we considered the ideal case with unconstrained signal lengths and general multidimensional code books. In this section we constrain our code books to be separable (i.e., we require that each sample of a signal be separately quantized). These watermarking systems will be referred to as *scalar* watermarking methods.

Quantization Index Modulation

A well-known example of a scalar watermarking method is referred to as least significant bit (LSB) watermarking. This method is most often applied to images where each pixel is represented by an 8-bit value. In the basic LSB watermarking method, each pixel is modified to carry 1 bit of information by changing the LSB to the required value. Since changing the LSB alters its value by 1 unit at most, the visual impact is typically minimal. Moreover, if the information being embedded is independent of the pixel's LSB values, on average only 50% of the pixels will be changed.

In this section we extend the LSB watermarking method to generalized scalar watermarking. The subcode books, $\mathcal{C}_m$, $m = 0, \ldots, M$, in M-ary scalar watermarking, are defined as multiples of a step size Δ:

$$\mathcal{C}_m = \{(m + kM)\Delta | k \in Z\}. \tag{5.40}$$

The union, $\mathcal{C}$, of the subcode books, $\mathcal{C}_m$, then consists of all multiples of Δ:

$$\mathcal{C} = \{k\Delta | k \in Z\}. \tag{5.41}$$

A message, m, is embedded in a sample value, s, by finding the closest value, x_m, in the subcode book, $\mathcal{C}_m$. This can be easily expressed using a uniform quantizer, Q, as:

$$x_m = Q_{M\Delta,\mathrm{m/M}}(\mathrm{s}), \tag{5.42}$$

where the uniform quantizer, Q_Δ, is defined as

$$Q_{\Delta,\delta}(s) = (\lfloor (s/\Delta) - \delta + 0.5 \rfloor + \delta)/\Delta, \tag{5.43}$$

and $\lfloor \mathrm{y} \rfloor$ denotes the floor operation.

Assuming a uniform distribution over the quantization interval (typically true if Δ is sufficiently small), then the embedding distortion, P_e, is seen to be equal to $P_\mathrm{e} = M^2\Delta^2/12$. Note that we are assuming real-value signals.

A received symbol, y, is decoded by finding the closest point in the set, $\mathcal{C}$, and determining (the syndrome of) the subcode book to which the point belongs. That is,

$$m = [\, y / \Delta] \bmod M. \tag{5.44}$$

where mod denotes the modup operation.

This method of scalar watermark embedding is called *quantization index modulation* (QIM). The name derives from the modulation (change) of the quantization index, $[s/\Delta]$, to the coset that corresponds to a chosen message, m, [76].

It is easily seen that for an additive uniform noise (AUN) channel with noise power less than $\Delta^2/12 = P_e/M^2$, the watermarking system is error free.

Distortion Compensated Quantization Index Modulation

The previous discussion focused on pure (scalar) quantization as an essential component of scalar quantization watermarking. However, as we have seen, in order to fully benefit from Costa's insight, we also need to apply distortion compensation. To apply Costa's method to the scalar case we proceed as follows. We fix a value α and quantize only a fraction α of a sample s:

$$\begin{aligned} s_m &= Q_{M\Delta,m/M}(\alpha s) + (1-\alpha)\, s \\ &= [(\alpha x - m\Delta)/M\Delta]M\Delta + m\Delta + (1-\alpha)s \\ &= [(x - m(\Delta/\alpha)/M(\Delta/\alpha)M(\Delta/\alpha)\alpha + m(\Delta/\alpha)\alpha + (\Delta/\alpha)(1-\alpha)s \\ &= \alpha Q_{M(\Delta\alpha),m/M}(s) + (1-\alpha)s \\ &= \alpha Q_{M\tilde{\Delta},m/M}(s) + (1-\alpha)s, \end{aligned} \tag{5.45}$$

where $\tilde{\Delta} = \Delta/\alpha$.

We observe that the scalar Costa method behaves similarly to the ideal Costa scheme:

- For an embedding distortion $P_e = M^2\Delta^2/12$, the sample values, s, are quantized using a step size, Δ/α. Without any additional measures this would lead to an enlarged embedding distortion, P_e/α^2 (since α is assumed to be smaller than 1).
- This enlarged distortion is *compensated* for by using only a fraction α of the quantized value and by retaining a fraction $(1-\alpha)$ of the original value. The quantity α is referred to as the *distortion compensation* parameter.

In the literature, scalar Costa watermarking is also referred to as *distortion compensated quantization index modulation* (DCQIM) [76] or the *scalar Costa scheme* (SCS) [125].

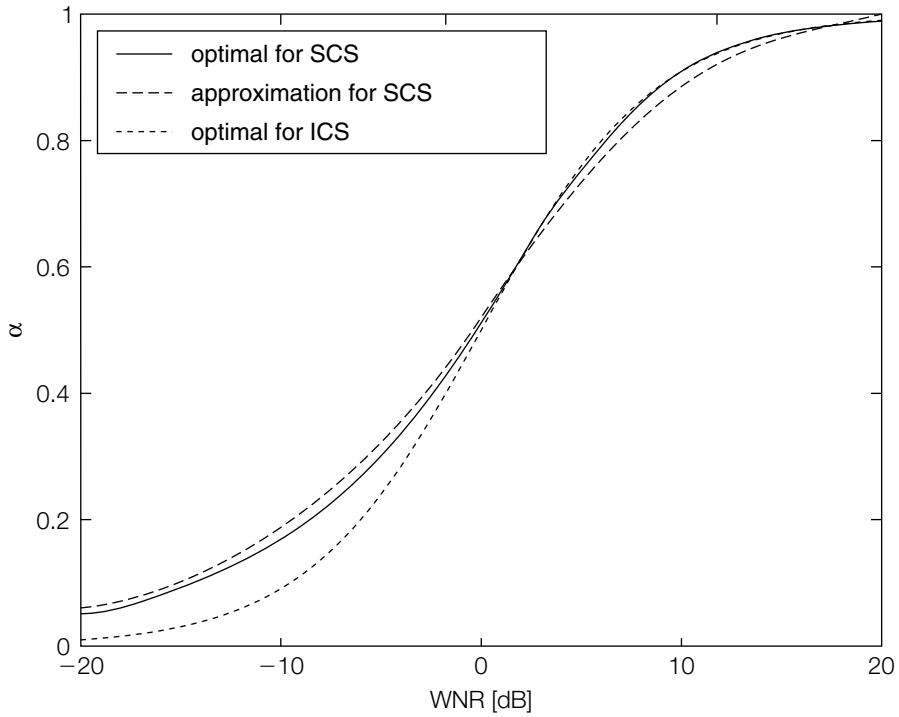


FIGURE 5.19

Optimal SCS α values for different WNR ratios (taken from Eggers *et al*. [125]) © [2000, 2005] IEEE.

Determining the optimal parameters and performance for SCS is analytically difficult. In [125], Eggers *et al*. studied the performance of SCS for the AWGN channel (see Figure 5.19). Eggers *et al*. concluded that in the high WNR region the optimal values for α^*_{SCS} and α^*_{ICS} differ very little. In the low WNR region, Eggers *et al*. experimentally found that α^*_{SCS} is approximated by:

$$\alpha^*_{\text{SCS}} = \frac{P_{\text{e}}}{P_{\text{e}} + 2.71P_{\text{a}}}. \tag{5.46}$$

Typical performance results are shown in Figure 5.20, where ICS, SCS, DM, and SS refer to the ideal Costa scheme, scalar Costa scheme, Dither modulation, and spreadspectrum, respectively. Dither modulation is an alternative name for quantization index modulation and refers to scalar dirty-paper coding without distortion compensation. Note that the *Watermark-to-Noise Ratio* (WNR) is defined as $P_{\text{e}}/P_{\text{a}}$. Similarly, the *Document-to-Watermark Ratio* (DWR) is defined as $P_{\text{s}}/P_{\text{e}}$. As expected, ICS performs better than SCS, and SCS performs better than DM. The spreadspectrum technique mentioned in this figure refers to an optimized

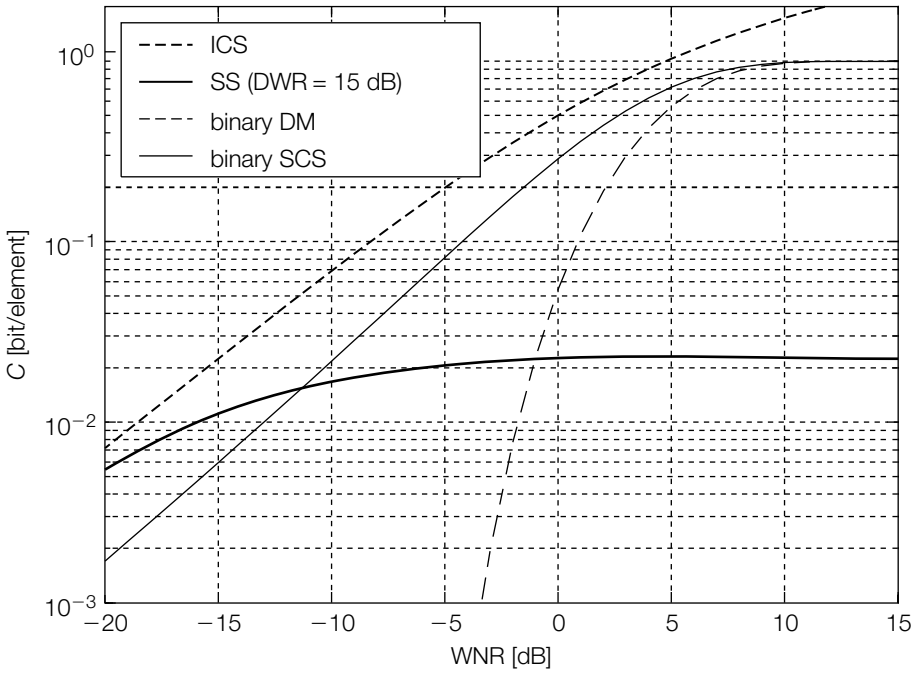


FIGURE 5.20

Performance of ICS, SCS, DM, and SS Eggers *et al*. (taken from [125]). © [2000, 2005] IEEE.

version of the E_BLIND/D_BLIND method of Chapter 3 and is discussed in more detail in Chapter 9. The spreadspectrum method ignores knowledge of the host signal other than its second-order statistics. The host signal is viewed as noise to the watermark signal and the capacity, R_{SS}, is given by Shannon's theorem:

$$R_{SS} = \frac{1}{2}\log\left(1 + \frac{P_e}{P_s + P_a}\right). \tag{5.47}$$

Eggers showed that over a large range of WNR values, binary SCS is a good approximation to ICS. In the low WNR region, spreadspectrum watermarking performs slightly better than SCS (but always less than ICS). In this same region, classical QIM completely fails, demonstrating that distortion compensation is an important ingredient in designing a robust watermarking system. In the high-WNR region, binary QIM and binary SCS differ very little. This is to be expected as in the high-WNR region, α^*_{SCS}, is close to 1.

Since the maximal rate for a binary scheme is limited to 1 bit per sample, ICS will outperform binary SCS for a sufficiently large WNR. The gap between

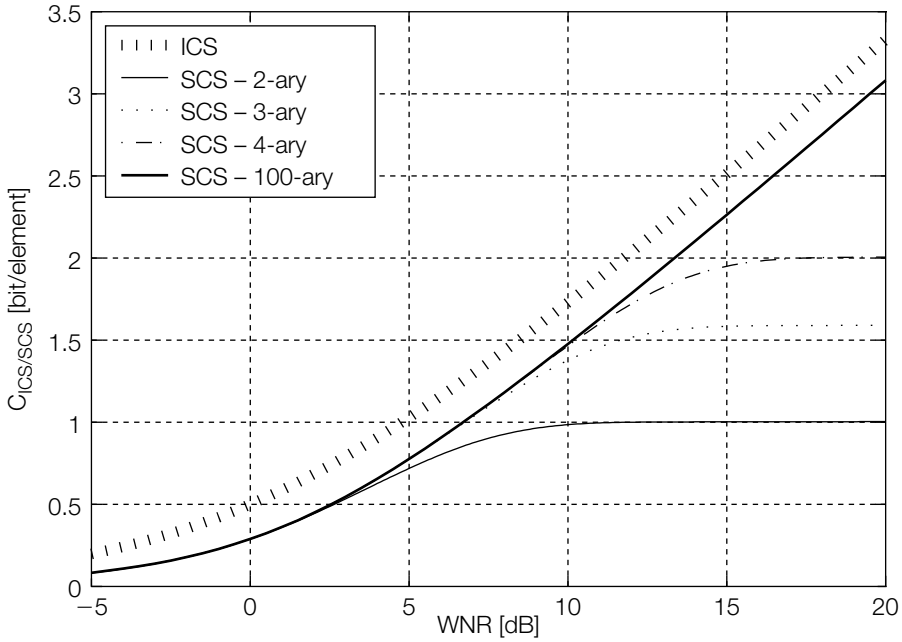


FIGURE 5.21

The capacity of *M*-ary SCS watermarking for an AWGN attack (taken from Eggers *et al*. [125]). © [2000, 2005] IEEE.

SCS and ICS can be moved to the right by switching to *M*-ary SCS. This is confirmed in Figure 5.21, which shows that the larger M is, the further to the right the *M*-ary system starts to taper off. However, also note that there is a systemic gap between ICS and *M*-ary SCS. This gap is a direct consequence of the fact that SCS is inherently one dimensional. This gap can only be overcome by switching to higher-dimensional vector quantizers.

5.3.4 Lattice Codes

In the previous section we observed that scalar watermarking systems have a systemic gap with respect to ICS. This gap can only be bridged by deploying higher-dimensional random codes. However, employing real random or pseudo-random coding is in general not practical as it involves finding the best matching code words in unstructured code books, which is known to be computationally very hard. In this section we therefore explore some structured higher-dimensional codes that may provide a better approximation to ICS than scalar codes.

We recall the definition of a dirty-paper code, $\mathcal{D} = (\mathcal{C}, \{\mathcal{C}_m\})$, as a set of points, $\mathcal{C}$ in $\mathcal{R}^n$, that is a disjoint union of subcode books, $\mathcal{C}_m$, $m \in \mathcal{M}$. The mapping that assigns the index, m, to a code word, $c \in \mathcal{C}_m$, is referred to as the *syndrome* mapping. The quantity, $\log(|\mathcal{M}|)/n$, is referred to as the rate, $R(\mathcal{D})$, of the dirty-paper code.

A *linear* dirty-paper code, $\mathcal{D} = (\mathcal{C}, \mathcal{C}_0)$, also referred to as a *lattice* dirty-paper code, is a dirty-paper code, $(\mathcal{C}, \{\mathcal{C}_m\})$, defined by the following properties:

- $\mathcal{C}$ and $\mathcal{C}_0$ are linear over the integers Z, that is, for any $c_1, c_2 \in \mathcal{C}$ respectively $\mathcal{C}_0$, and any integers a_1, a_2, the linear combination $a_1c_1 + a_2c_2$ is also in $\mathcal{C}$ and $\mathcal{C}_0$, respectively.
- $\mathcal{C}$ is generated by a finite number of elements.
- $\mathcal{C}_0 \subset \mathcal{C}$. The codes $\mathcal{C}_0$ and $\mathcal{C}$ are referred to as the inner and the outer code, respectively.
- A coset is defined as a translated version $c + \mathcal{C}_0 \subset \mathcal{C}$ of the inner code $\mathcal{C}_0$, $c \in \mathcal{C}$.
- The set of cosets is finite.
- The set of cosets is defined as the set of subcode books of the dirty-paper code.
- The set of messages, $\mathcal{M}$, is an index set for the set of cosets.

In the previous section on scalar watermarking we came across a linear code in our construction of the scalar watermarking systems, where the sets $\mathcal{C}$ and $\mathcal{C}_0$ consisted of multiples of Δ and $M\Delta$, respectively. Examples of multidimensional dirty-paper codes can be easily constructed as Cartesian products of one-dimensional codes, but for coding purposes these product codes obviously do not offer any gain over one-dimensional codes. An example of a rate 0.5, nonseparable, linear dirty-paper code is provided by the *quincunx* dirty-paper code (see Figure 5.22).

- The outer code $\mathcal{C}$, is the Cartesian product $\mathrm{Z} \times \mathrm{Z}$ as a subset of the Cartesian product $\mathcal{R} \times \mathcal{R}$.
- The inner code, $\mathcal{C}_0$, is defined as the set of points in $\mathcal{C}$ with the sum of their coordinates even, that is, $\mathcal{C}_0 = \{(k_1, k_2) \in \mathcal{C}\}$ where $(k_1 + k_2)\%2 = 0$.

This quincunx code has two cosets, with $\mathcal{C}_1$ given by $\mathcal{C}_1 = \{(k_1, k_2) \in \mathcal{C} : k_1 + k_2\%2 = 1\}$.

As in the one-dimensional case, lattice codes are easily extended to include distortion compensation. As distortion compensation scales all subcode books by the same factor, the linear code property is maintained.

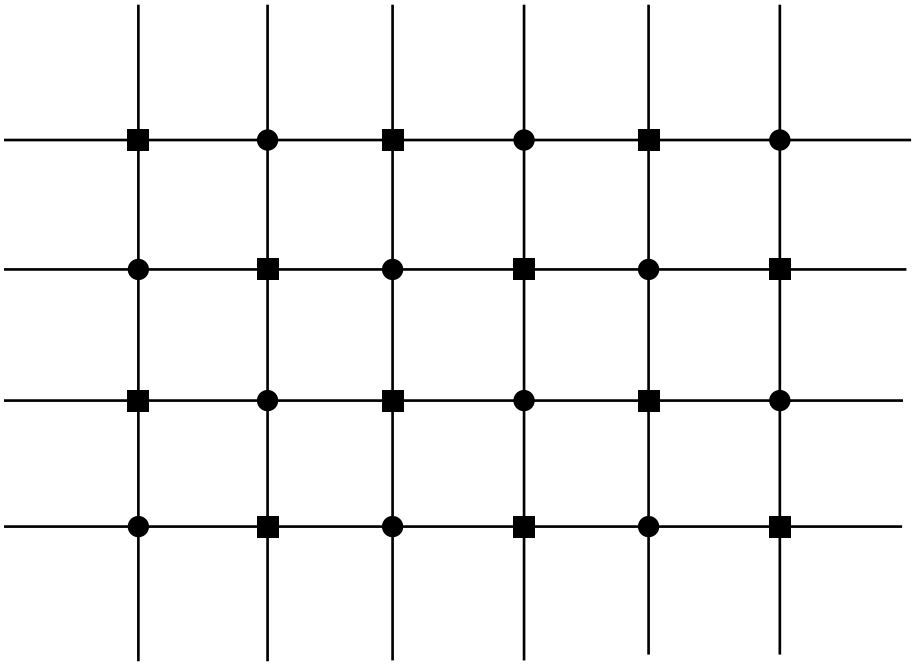


FIGURE 5.22

The quincunx lattice code, where the squares and the circles are denoting the two cosets.

5.4 SUMMARY

This chapter has explored the potential impact of side information on the performance of watermarking systems. The main points made here include the following.

- During watermark embedding, the cover Work is known to the embedder and can be treated as side information about the communication channel rather than as unknown noise.
- The problem of finding an optimal embedding scheme given a fixed detector is introduced. This can be viewed as an optimization problem:
  - Maximize robustness while maintaining a constant fidelity.
  - Maximize fidelity while maintaining a constant robustness.
- Although in linear correlation systems the detection statistic can serve as an estimate of robustness, this is not effective for normalized correlation systems. In such systems, we must optimize with respect to a more

explicit estimate of robustness, such as the amount of white noise that may be added before the Work is expected to reach the edge of the detection region.

- *Informed coding* refers to the use of side information during the selection of a message mark. This can be done by defining a *dirty-paper code* in which each message is represented by several alternative code words.
- Coding for Gaussian channels allows a geometrical interpretation.
- The special case of optimal watermarking of the all-zero signal is covered by the Shannon theorem on AWGN channels.
- Costa's theorem on optimal watermarking is stated and provided with a sketch of a geometrical proof.
- The two main tools of optimal watermarking are dirty-paper codes and distortion compensation.
- Distortion compensated quantization index modulation (DCQIM) is a special case of dirty-paper coding with distortion compensation applied to scalar and finite dimensional signals.
- Lattice codes are a special case of multidimensional dirty-paper codes that are able to approach the performance of the ideal Costa method.

CHAPTER 6

Practical Dirty-Paper Codes

Random codes, such as those employed in the E_DIRTY_PAPER/D_DIRTY_PAPER system or the ones used in the proof of Costa's theorem, have a high likelihood of producing good performance. Unfortunately, they are not generally practical. Both the encoder and the decoder are required to find the closest code word to a given vector, and for large, purely random codes, this requires prohibitive computational time and storage. To realize the large payloads that Costa's theorem says should be possible, we must use codes structured in a manner that allows efficient search for the closest code word to a given vector. This chapter discusses some of the research that has been devoted to the problem of constructing such codes.

We begin in Section 6.1 with a discussion of the desirable characteristics of practical codes. This is followed in Section 6.2 with a brief rundown of various approaches that have been used to construct codes with these characteristics. The rest of the chapter is devoted to the approach of dirty-paper code design most widely used in watermarking: *quantization index modulation* (QIM). Section 6.3 introduces the basics of *lattice coding*, which is the most common implementation of QIM. Sections 6.4 and 6.5 delve deeper into the topic of lattice coding by discussing, respectively, common tricks used in implementing them and various types of lattices that are available. Section 6.6 then addresses one of the biggest practical problems with using lattice codes for watermarking: their fragility against simple valumetric scaling (e.g., changing the brightness of an image or the audio volume of music). Finally, Section 6.7 describes dirty-paper *trellis codes*, which constitute a nonlattice implementation of QIM using trellis quantizers.

6.1 PRACTICAL CONSIDERATIONS FOR DIRTY-PAPER CODES

Whatever the application of a dirty-paper code, there are three functions that must be accomplished efficiently: (1) the encoder must be able to identify the set of code signals that represent a message and find the best one among them

for transmission, given its side information; (2) the decoder must be able to find the code signal that is closest to a received signal, thereby finding the message that was most likely encoded; and, of course, (3) we want a code that performs well, in the sense of achieving something close to the theoretical capacity of the channel defined by the application.

Each of these three properties of a dirty-paper code—computationally efficient encoding, computationally efficient decoding, and performance—has nuances that vary from application to application. These are discussed below.

6.1.1 Efficient Encoding Algorithms

The task of the encoder is to find the best signal for encoding a given message in the context of given side information. This begs the question: What is meant by "best"? The answer, of course, depends on the application.

In the pure, theoretical dirty-paper channel, we are concerned only with ensuring that we do not violate the power constraint. Thus, the transmitted vector, $\mathbf{x}$, must satisfy

$$\frac{1}{N}\sum_{i}^{N}\mathbf{x}[i]^2 \leq p. \tag{6.1}$$

If we plan to compute $\mathbf{x} = \alpha(\mathbf{u} - \mathbf{s})$, where α is a constant, $\mathbf{u}$ is the code word, and $\mathbf{s}$ is the first noise source (the dirty paper), then we need only choose any $\mathbf{u}$ that results in an $\mathbf{x}$ of sufficiently low power. If more than one is available, the choice can be arbitrary. If none is satisfactory, then we simply fail to transmit the message. Basically, we seek any $\mathbf{u}$ that lies within a sphere of radius $\alpha^{-1}\sqrt{p}$ around $\mathbf{s}$.

In real applications, the criteria are often quite different from a fixed power constraint. To begin with, in an application where the power of $\mathbf{x}$ is our main concern, we might prefer a lower-power $\mathbf{x}$ over a higher-power one, even if both lie within the power constraint. And if no $\mathbf{u}$ can be found that satisfies the power constraint, we might want to relax that constraint so that we guarantee transmission of the message. Alternatively, we might be prepared to reduce α, resulting in a lower—but at least nonzero—probability of successfully transmitting the message. In any of these cases, the best code signal, $\mathbf{u}$, is simply the one that is closest, in a Euclidian sense, to the first noise source, $\mathbf{s}$. The sphere of radius $\alpha^{-1}\sqrt{p}$ does not enter into the search (though it may enter into our subsequent decision of what to do with the $\mathbf{u}$ that we find).

Further afield from coding for the pure Gaussian dirty-paper channel are applications in which a mean square type of *power* is not the most important consideration. For example, in watermarking, where $\mathbf{x}$ corresponds to the watermark pattern added to the cover work, the most important issue is the perceptual distance between the original and the marked Work. As discussed in Chapter 8, this distance is not well estimated by measuring a mean squared error. Instead, we would like to use more sophisticated perceptual models.

Thus, the best **u** is the one that, once embedded in the cover Work, will result in the least perceptual distance as measured by some model.

In steganography (see Chapter 12), we are concerned with more than just perceptual issues. Although we need to choose a **u** that results in a stego-Work that is perceptually natural, we also need to ensure that embedding the code signal does not change the Work's statistics in a way that can be detected by various sophisticated tests. In general, the best **u** will be the one that results in the least suspicious Work, according to some model of suspicion.

In general, then, there is a wide variety of criteria that might be used for deciding which encoding of a given message is best. One way to formalize the range of possibilities is to say that, given an application and a method of transmitting the selected code signal, we have a function $c = C(\mathbf{u}, \mathbf{s})$ that specifies a *cost* of transmitting **u** under the conditions described by **s**. This cost might be, for example, a measure of perceptual distortion, a measure of suspicion, or a measure of the likelihood that the message will be incorrectly decoded. The task of the encoding algorithm, then, is to find the lowest-cost code signal for the desired message.

Different types of dirty-paper codes provide efficient search algorithms for different sets of cost functions. Ideally, we would like to use an algorithm that perfectly matches the cost function appropriate for our application. In practice, however, we often have to make do with a cost function that only approximates the one we want. For example, some codes provide no good algorithm to perform a search for the **u** that is perceptually closest to **s**, so we must make do with a search based on mean squared error (MSE).

6.1.2 Efficient Decoding Algorithms

When it receives a signal that might have been distorted, the decoder must be able to efficiently find the code word that was most likely transmitted. This means finding the closest code word to the received signal, according to some measure of distance that is related to the types of distortion expected. The measure used is often referred to as the *decoding metric*, although it is not always a metric in the strict mathematical sense.

Because different applications entail different types of distortion, they are best implemented with different decoding metrics. Just as we would ideally like an encoding search algorithm that perfectly matches the right cost function for our application, we would ideally like a decoding search algorithm that matches the right decoding metric. Again, however, not all codes provide efficient search algorithms for every possibility.

Traditionally, the search for the most likely symbol is split into a real-valued and an algebraic phase, respectively. In the former phase, the search space for the most likely code word is restricted to some discrete set of code words. In the second algebraic phase, typically referred to as *error correction decoding*, the discrete search space is reduced to a single code word—the most likely

code word. In more advanced decoding schemes, some information from the real-valued phase is retained in the algebraic phase; such systems are referred to as *soft decoding schemes*.

Most conventional error correction codes are defined for binary code words that must be modulated in some manner to produce real-valued watermark patterns (see Chapter 4). These typically allow efficient search for the closest code word in terms of Hamming distance. In watermarking most media, however, Hamming distance is not the most appropriate metric to use, and most of the dirty-paper codes described in this chapter are not designed for it (though dirty-paper trellis codes can be configured for this metric—see Section 6.7).

Most of the dirty-paper codes described here are defined for code words in real-valued vector spaces. The most common codes—lattice codes—allow very efficient search for the closest code words according to Euclidian distance. This is appropriate when we expect the transmitted signals to be distorted by additive white Gaussian noise. Unfortunately, in many watermarking applications, this is not the most likely type of distortion. As will be discussed in Chapter 9, more likely distortions include quantization due to lossy compression and "valumetric" scaling such as changing the brightness of images or the volume of audio signals. Though the former of these two can be reasonably modeled as additive noise (see Section B.5 in Appendix B), valumetric scaling cannot, and lattice codes are thus highly susceptible to this type of distortion.

A better decoding metric in the face of valumetric scaling is angle distance, which can be applied by finding the code signal that has the highest normalized correlation with the received signal. With simple lattice codes, this metric is more difficult to employ. Note, however, that the ideal Costa scheme employs code words of equal norm and is therefore suitable for decoding with angle distance.

In real applications, as opposed to the ideal Costa scheme, the best decoding metric is often different from the best encoding cost function. In watermarking, the best encoding cost function depends on perceptual characteristics of the embedding process, while the best decoding metric depends on the types of distortions we expect media to undergo. In steganography, the best encoding cost function depends on statistical characteristics of the media, and we often do not need to worry about the subsequent distortions. These applications are thus quite different in this respect from Costa's theoretical dirty-paper channel, where Euclidian distance is both a good encoding cost function (because the limitation on the encoder is expressed as a power constraint) and a good decoding metric (because the noise in the channel is additive white Gaussian).

6.1.3 Tradeoff between Robustness and Encoding Cost

A third critical consideration, apart from the existence of high-speed coding and decoding algorithms, is how well a code actually performs. What is the typical cost of the best encoding for a typical message with typical side information?

How likely is a message to be correctly received after the transmitted signal is distorted? These properties of a code are decided by how it arranges code words in the space of possible signals and how well distortion compensation can be applied.

Theoretical proofs of channel capacity like Costa's (Section 5.3.2) rely on the fact that random codes tend to exhibit good properties as the number of dimensions increases toward infinity. Practical codes, however, must arrange their code signals into structures that allow for efficient search, and these structures often do not behave as well as their random, theoretical counterparts. That is, they often do not achieve capacity.

In conventional coding, performance depends only on how well code words are separated from one another, which can be expressed in terms of *code separation* or *sphere packing* (see Chapter 5). This determines the robustness of a code (i.e., how likely messages are to survive transmission). In dirty-paper codes, however, we are additionally concerned with how code words are divided into cosets. This determines the typical cost of encoding a message.

The two attributes of a dirty-paper code—code separation and coset formation—are fundamentally at odds with one another. For robust encoding, we want code words for different messages to be widely spaced so that noise doesn't move a code word into the detection region for another message (the $I(Y; U_\alpha)$ part of Equation 5.33). For low encoding cost, we want code words for different messages to be *closely* spaced, so that we are likely to find an encoding of any given message near any given side information signal (the $I(S; U_\alpha)$ part of Equation 5.33). Finally, a dirty-paper code should be amenable to distortion compensation to allow balancing the opposing encoding and decoding attributes.

Various solutions to this tradeoff are illustrated in Figure 6.1. Here we show a space of possible signals consisting of two integers, and indicate various ways a decoder might decode each such signal into one of four messages. Figure 6.1(a) shows a conventional code, in which the detection region for each message is a single, contiguous shape. This code uses all the available redundancy for robustness, and might be appropriate for an application with high noise but a loose constraint on encoding cost (e.g., a large value of p for the power constraint in Costa's dirty-paper channel). At the other extreme, Figure 6.1(c) shows a "maximally" dirty-paper code, in which all the redundancy is used for maintaining low encoding cost, and none of it is used for robustness. This would be appropriate for an application with a tight constraint on encoding cost (small value of p in Costa's channel) but essentially zero noise.

Of course, we would ideally like a code that provides the tradeoff between robustness and encoding cost most appropriate for our application. Best would be a family of codes in which we could determine this tradeoff by choosing one or a small number of parameters. Lattice codes and dirty-paper trellis codes provide such a capability (though with dirty-paper trellis codes it is tricky to control).

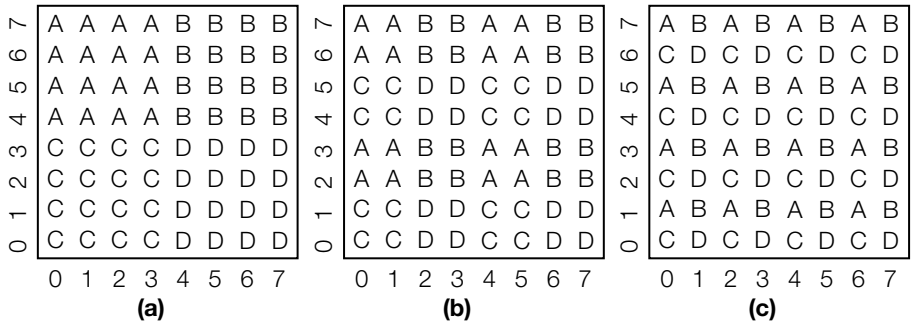


FIGURE 6.1

Various configurations of a four-message ("A," "B," "C," or "D") code that represent different tradeoffs between robustness to noise and reduction of encoding cost.

6.2 BROAD APPROACHES TO DIRTY-PAPER CODE DESIGN

Before getting into specific coding methods, it is useful to understand the broad approaches that have been employed in designing them.

6.2.1 Direct Binning

The most straightforward way to construct a dirty-paper code is to explicitly define the set of code words and their division into cosets (or *bins*). For lack of a better term, we shall refer to this as *direct binning*.

Direct binning has also been studied using orthogonal and quasi-orthogonal codes [12]. These studies showed, both experimentally and theoretically, that such dirty-paper codes can produce good results when the document-to-watermark ratio (DWR) is large.

Unfortunately, as pointed out at the beginning of this chapter, this strategy leads to computationally expensive encoding and decoding. Encoding a message requires an exhaustive search through all the signals in *that* message's coset. Decoding a message requires an exhaustive search through *all* the messages' cosets. Unless the size of each coset and the number of possible messages are small, these searches can be prohibitive.

6.2.2 Quantization Index Modulation

Perhaps the broadest approach to making practical dirty-paper codes is known as *quantization index modulation* (QIM) introduced by Chen and Wornell [76] and described in Sections 5.3.3 and 5.3.4. The basic strategy is to define a family of quantizers, $\mathcal{Q}$. Each quantizer, $Q_m \in \mathcal{Q}$, is a procedure for mapping any member of the set of all vectors in marking space to the closest of a smaller set of points, $\mathcal{U}_m$. Different messages are associated with different

quantizers, and the sets of points to which they quantize are the corresponding signal cosets in the code.

The embedder applies the quantizer for the desired message to the vector for the cover Work, finding the closest point in that quantizer's range. It then modifies the cover Work so that the watermarked work will lie on or close to that point.

$$\mathbf{c_w} = E(\mathbf{c}, m) = Q_m(\mathbf{c}). \tag{6.2}$$

The decoder applies all the quantizers and identifies the one that contains the closest point.

$$\hat{m} = D(\mathbf{c_{wn}}) = \underset{\mathbf{m}}{\operatorname{argmin}} |Q_m(\mathbf{c_{wn}}) - \mathbf{c_{wn}}|. \tag{6.3}$$

QIM provides a basic recipe for making a practical code from known quantization procedures. Given that many quantizers are very efficient, this recipe can produce very efficient codes. For example, a simple linear lattice quantizer quantizes to cosets of infinite size in constant time. This means our encoder can take a trivial amount of time.

The decoder, at worst, must apply all possible quantizers, meaning that the code may be impractical if the number of messages is large. However, it is sometimes possible to create a single quantizer that quantizes to the union of all the cosets, and provides enough information about the resulting point to determine which coset originally contained it. This can make the decoder as efficient, or almost as efficient, as the encoder.

6.2.3 Dither Modulation

Dither modulation (DM), also first described by Chen and Wornell [76], is a general trick for building QIM systems by combining any type of quantizer with any conventional (non-dirty-paper) message coding scheme to obtain a QIM code. The idea is to start with a base quantizer, Q_{base}, and produce our family of message-specific quantizers by offsetting Q_{base} with a message-specific shift, or *dither* (see Section 5.3.4). DM is a powerful technique that, in principle, can be used to define a wide variety of different systems by using different quantizers and different conventional message codes. In practice, however, it is typically applied with simple rectilinear lattice quantizers and simple orthogonal codes.

6.3 IMPLEMENTING DM WITH A SIMPLE LATTICE CODE

The majority of research in watermarking with practical dirty-paper codes has focused on applying QIM and DM with regular lattice quantizers (see, for example, [77–79]). These codes can be very easy to implement and are robust against additive white Gaussian distortion [299].

We begin our discussion of lattice codes by constructing a simple example. This example is essentially the same as the system first proposed by Chen

and Wornell [77, 78]. A very similar system has been proposed by Eggers *et al*. under the name *scalar Costa scheme* (SCS) [125]. Variations of this idea appeared earlier, such as watermarking by quantizing transform coefficients [297] and embedding in the least significant bits (LSBs) of individual pixels. Several variations on this basic design are reviewed in Chen and Wornell [78, 79].

The simplest N-dimensional lattice consists of all integer linear combinations of a set of N orthogonal vectors $\{\mathbf{w}_{\mathbf{r}1}, \mathbf{w}_{\mathbf{r}2}, \ldots, \mathbf{w}_{\mathbf{r}N}\}$. The simplest dirty-paper code on such a Cartesian lattice is constructed by taking as the zero sublattice all the integer linear combinations of $\{2\mathbf{w}_{\mathbf{r}1}, 2\mathbf{w}_{\mathbf{r}2}, \ldots, 2\mathbf{w}_{\mathbf{r}N}\}$ (see Section 5.3.4). The number of cosets (i.e., messages), is easily seen to be equal to 2^N. Each coset can be represented by an N-dimensional binary vector $\mathbf{b} = (b_1, \ldots, b_N)$ such that that the coset corresponding to $\mathbf{b}$ consists of the set of vectors of the form $\sum(b_i + 2k_i)\mathbf{w}_{\mathbf{r}i}$, where b_i is a component of $\mathbf{b}$ and k_i is an integer.

For a given vector, $\mathbf{v}$, at encoding time the closest code word, $\mathbf{z}_\mathbf{z}$, in a coset for vector $\mathbf{b}$ can be easily found, component by component. The first step is to project to a basis vector:

$$p[i] = \frac{\mathbf{v} \cdot \mathbf{w}_{\mathbf{r}i}}{|\mathbf{w}_{\mathbf{r}i}|^2}. \tag{6.4}$$

In a second step we subtract b_i from $p[i]$, round to the nearest even integer, and add b_i again.

$$q[i] = 2 * \left\lfloor \frac{p[i] - b_i + 1}{2} \right\rfloor + b_i - 1. \tag{6.5}$$

Here, b_i is the message dependent dither. Finally, we find $\mathbf{z}_\mathbf{z}$ as

$$\mathbf{z}_\mathbf{z} = \sum_i q[i]\mathbf{w}_{\mathbf{r}i}. \tag{6.6}$$

At decoding time, a vector, $\mathbf{v}$, is first projected to the basis vectors as in Equation 6.4. For each component, the projected value is rounded to the nearest integer, $q[i]$:

$$q[i] = \lfloor p[i] + 0.5 \rfloor. \tag{6.7}$$

Finally, the coset vector, $\mathbf{b}$, is easily found as the vector of parities $\mathbf{b} = (q[1] \mod 2, \ldots, q[N] \mod 2)$. Note that an orthogonal lattice code as described above can be viewed as a sequence of scalar codes (without distortion compensation; see Section 5.3.3).

INVESTIGATION

Watermarking with an Orthogonal Lattice Code

To illustrate the use of lattice codes, we now present a simple example using an orthogonal lattice code directly in media space. The system we present here is

based on that proposed by Chen and Wornell [79] and is related to an earlier system proposed in Swanson *et al.* [396].

System 9: E_LATTICE/D_LATTICE

The E_LATTICE/D_LATTICE watermarking system embeds 1 bit per 256 pixels in an image. As our test images are $240 \times 368 = 88{,}320$ pixels, this system embeds $88{,}320/256 = 345$ bits in each image.

In the E_LATTICE embedder, the 345 message bits are first encoded with the trellis code of Chapter 4, reproduced in this chapter as Figures 6.2 and 6.3. This produces a sequence of 1,380 coded bits (the algebraic part of the embedding process). Embedding these bits with the lattice coding system described above requires correlating the image against 1,380 orthogonal reference marks. If the

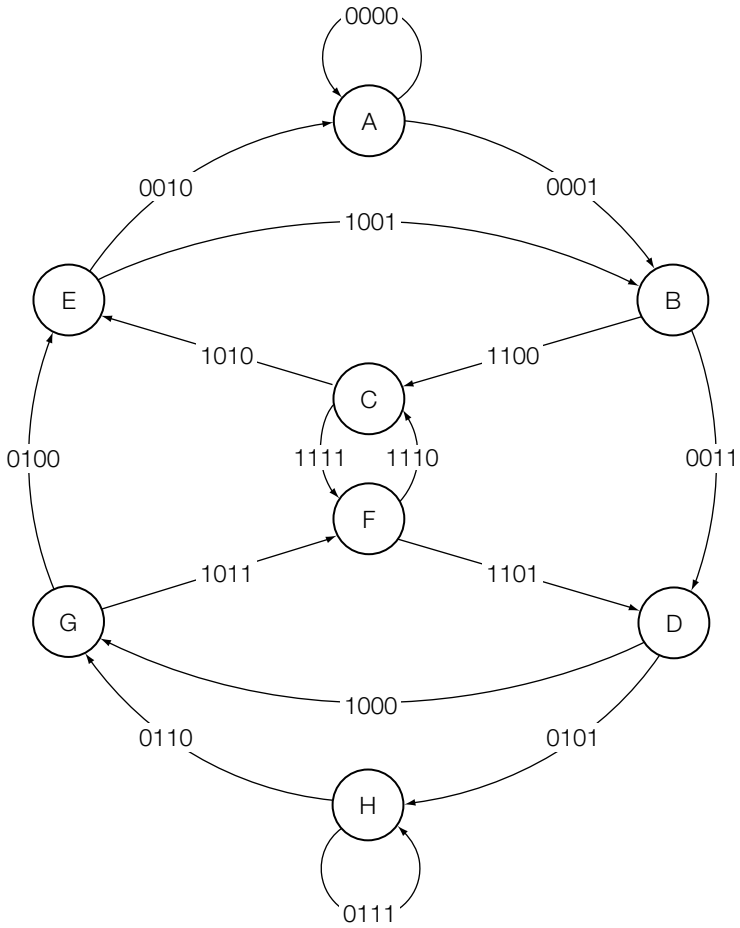


FIGURE 6.2

An eight-state convolutional code.

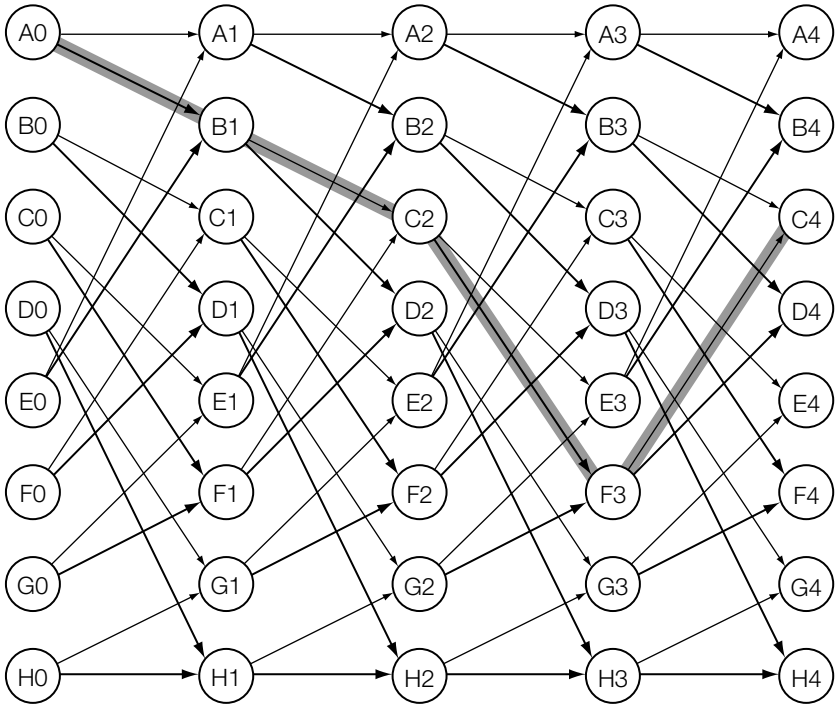


FIGURE 6.3

Trellis representation of the code in Figure 6.2. Each row corresponds to one of the eight states. Each column corresponds to an iteration of the encoding (column 0 is before encoding starts, column 1 is after encoding the first bit, and so on). The labels along the arcs have been removed for clarity. The highlighted path corresponds to the encoding of 1010.

reference marks are nonzero everywhere in the image, these correlations are costly. To make the system more efficient, we use a simple form of spatial sequencing, dividing the image into 8×8 blocks and embedding 1 bit into each. Each bit is embedded by correlating a block against a single 8×8 reference pattern, $\mathbf{w_r}$, and quantizing the result to an odd or even integer. Essentially we are restricting our reference patterns to have limited support, viz., only to an 8×8 block.

Thus, the exact steps of the E_LATTICE embedding algorithm are as follows:

1. Encode the message into a sequence of coded bits, $\mathbf{m}[1], \mathbf{m}[2], \ldots, \mathbf{m}[1380]$.
2. Divide the cover image, $\mathbf{c_o}$, into 1,380 8×8 blocks.
3. Modify each block to embed a corresponding bit.

After adding all the vectors, $\mathbf{w}_{a\,i}$, to their respective blocks, $\mathbf{c}_i$, we have the watermarked image.

The D_LATTICE detector is simpler than the embedder. At each block, we compute $\mathbf{z}[i]$ given in Equations 6.4 and 6.7. From this, we then detect each bit of the coded message. Bit i of the coded message is 1 if $\mathbf{z}[i]$ is odd, and 0 if it is even. This coded message is then decoded with a Viterbi decoder to yield 345 message bits. The detector does not attempt to determine whether or not a watermark was actually embedded.

Experiments

To test the performance of this system, we embedded random messages in 2,000 images using the same reference mark used for our earlier experiments on block-based watermarking systems. These parameters result in an average MSE or an expected distortion per pixel (root MSE) of about 2.

A different message was embedded in each image. We then ran the D_LATTICE detector and compared the detected messages bit-for-bit with the embedded messages. It is possible, due to round-off and clipping errors, for some bits to be incorrectly embedded. This occurred in about 0.8% of the images, so we have an embedding effectiveness of 99.2%.

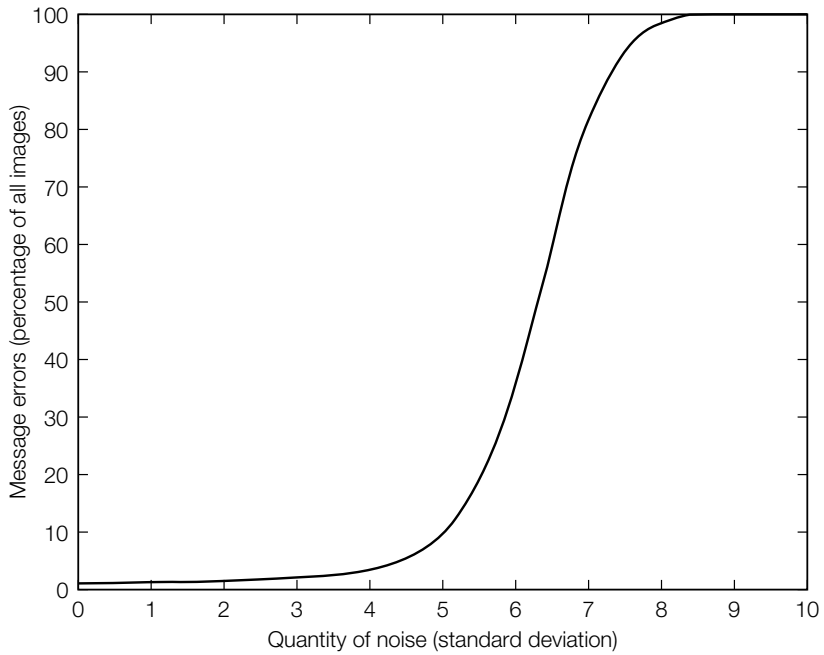


FIGURE 6.4

Effect of additive white Gaussian noise on E_LATTICE/ D_LATTICE watermarking system.

The logic behind a lattice-based watermark assumes additive distortion in marking space. We thus tested the system for robustness against additive white Gaussian noise. Noise of different amplitudes was added to each image, and the D_LATTICE detector was applied. Figure 6.4 plots the message error rate as a function of noise amplitude. This shows that, with our parameter settings, 90% of the watermarks survived noise amplitudes of over 5.

6.4 TYPICAL TRICKS IN IMPLEMENTING LATTICE CODES

Though the system described above is very simple, it is not actually the simplest lattice code that has been tried. That honor probably goes to the practice of embedding watermarks in the LSBs of the cover Work's sample values.[1] Of course, this practice results in very fragile watermarks, so it is primarily used for content authentication (Chapter 11), where fragility is an asset, and steganography (Chapter 12), where we often do not expect any distortion.

To make a robust watermark, like the E_LATTICE/D_LATTICE system, we needed to employ some straightforward tricks. First, we used a different lattice than would appear in LSB watermarking. Next, we employed a trick known as *spreading functions*. These two topics are discussed below, together with two tricks we did not employ: *distortion compensation* and *dither*.

6.4.1 Choice of Lattice

All lattices are equal but some are more equal than others. All lattices are equal in the sense that each lattice is simply a representation of Z^n in R^n. However, some lattices better resemble a random distribution of points than others. Recalling the proof of Costa's theorem, we expect a better coding performance for lattices that look more random. As it turns out, the simple lattice code of the previous example is not particularly optimal. Better lattices will be discussed in Section 6.5.

6.4.2 Distortion Compensation

Recall that distortion compensation for a lattice Λ is expressed as follows (Equation 5.31):

$$s_m = \alpha Q_{\Lambda/\alpha,m}(s) + (1 - \alpha)s, \tag{6.8}$$

[1] In the case of LSB watermarking, the quantization process is not explicitly included in the watermarking algorithm. Rather, the algorithm exploits the quantization implicit in using integers to represent real-world values, such as pixel intensities or audio samples.

where m is a coset representing a message and α is the distortion compensation parameter. We can write this formula in a slightly different way as

$$s_m - s = \alpha(Q_{\Lambda/\alpha,m}(s) - s). \tag{6.9}$$

In other words, the difference between the marked signal and the original signal is equal to a scaled version of the quantization error for the coarser lattice Λ/α. We can now generalize to

$$s_m - s = \alpha(Q_{\beta\Lambda,m}(s) - s), \tag{6.10}$$

where β is an additional degree of freedom. For a given distortion and a fixed lattice type, the Costa theorem requires $\alpha\beta = 1$. However, for finite lattices and non-Gaussian signals this is not necessarily the optimal choice.

6.4.3 **Spreading Functions**

The idea of using spreading functions [76] is to "spread" each dimension of the lattice over several samples and frequencies in the cover Work. In E_LATTICE/D_LATTICE, we spread each dimension into an 8×8 block by projecting the block onto an arbitrary reference pattern, $\mathbf{w_r}$.

In contrast, in LSB watermarking each dimension of the lattice corresponds to one sample of the cover Work, and in several proposed systems each dimension corresponds to one term in a frequency transform. If that sample or frequency is subsequently distorted by normal processing or intentional attack, the bit it encodes will likely be corrupted.

When we employ spreading functions, we gain an advantage of spread spectrum watermarking—the system is robust against uncorrelated noise added to the values over which the watermark has been spread.

6.4.4 **Dither**

There is one more trick commonly used in lattice-coded watermarking, though we did not use it in the E_LATTICE/D_LATTICE system. This is usually referred to as *dithering*, and unfortunately can be confused with the message-dependent "dither" in the namesake of DM. In general, it will be clear from the context which type of dither is being referred to.

The problem addressed by dither is that the histogram of a typical signal quantized by a fixed scalar quantizer is statistically quite different from the original histogram (see Figure 6.5). One consequence is that the difference between the original signal and the watermarked signal might be perceived as correlated noise, which in general results in lower perceptual quality.

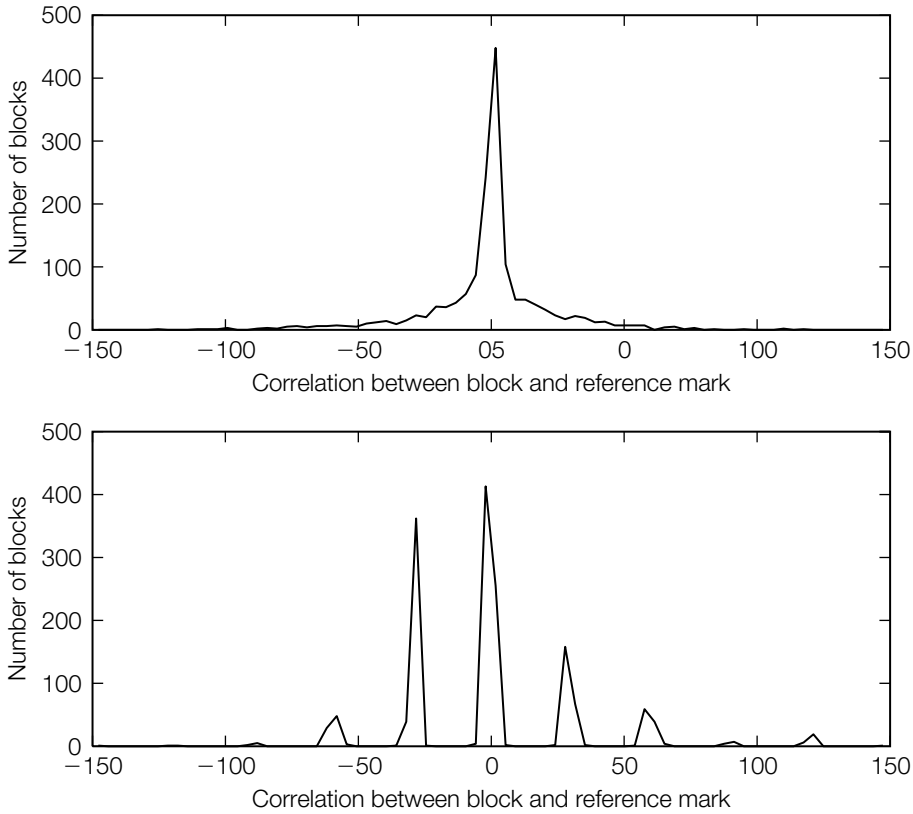


FIGURE 6.5

Histograms of correlations between image blocks and reference marks. The image was watermarked with the E_LATTICE embedder. The top graph shows the histogram of correlations with the wrong reference mark. The bottom shows correlations with the correct reference mark, and illustrates the spikes that result from quantization. By looking for these spikes, an attacker can obtain valuable information about the reference mark.

To solve this problem, we can "dither" the lattice by adding an additional, *key*-dependent (and message-*independent*) offset. Thus, the watermark embedding function becomes

$$E(\mathbf{c}, m) = \mathbf{c} + Q_{\text{base}}(\mathbf{c} - C(m) - J(k)) + C(m) + J(k), \tag{6.11}$$

where $J(k)$ is a pseudo-random function of the key, k, and $C(m)$ is a message-dependent dither (see Section 6.5.3). The same dither function is then used in the decoder. Without knowing the key, and hence this dither signal, the adversary cannot search for the telltale spiked histograms that indicate a correct guess at the spreading function.

6.5 CODING WITH BETTER LATTICES

In discussions of lattice coding, it is often commented that better results might be obtained using lattices other than the simple one used above. We therefore turn our attention to three basic questions. First, how can we describe and use other lattices? Second, what criteria should we consider when choosing a lattice? And third, what vectors should we use for dithering when implementing DM with a given lattice; that is, what should the message-dependent dither, $C(m)$, be?

We then conclude this section by implementing a new watermarking system that uses the eight-dimensional E_8 lattice, which was first applied to DM watermarking by Zhang and Boston [467]. We find, as they did, that this yields watermarks that are robust to more noise than those obtained with an orthogonal lattice, for the same embedding distortion.

6.5.1 Using Nonorthogonal Lattices

Formally, a lattice is defined as a set of vectors, $\Lambda \subset \mathcal{R}^n$, such that

1. Λ is a linear module over the integers, that is, for every integer a and b and for every $\mathbf{v}$ and $\mathbf{w}$ in Λ, the vector $a\mathbf{v} + b\mathbf{w}$ is also in Λ.
2. Λ has dimension n, that is, Λ contains a set of n vectors $\mathcal{B} = \{\mathbf{v}_i\}_{i=1}^n$ such that (1) every element of Λ is an integer linear combination of elements in $\mathcal{B}$, and (2) the set $\mathcal{B}$ is a basis of the real vector space $\mathcal{R}^n$. Such a set $\mathcal{B}$ is referred to as a basis of Λ.

As a consequence of the definition above, every lattice can be represented by an $n \times n$ matrix B where the columns of B form a basis of Λ.

For each lattice, Λ, we can also define its *dual lattice*, Λ^*, as the set of all vectors, $\mathbf{x}$, that have integer correlation with every vector in Λ. It is easily verified that the set Λ^* is linear over the integers. Moreover, one can easily verify that if B is a generating matrix of Λ, then B^{-t} (the inverse transpose of B) is a generating matrix of Λ^* and therefore that Λ^* satisfies the rank condition.[2]

A wide variety of different lattices have been identified and studied, each with its own set of basis vectors. The reader is directed to Conway and Sloane [84] for a comprehensive discussion of lattice theory. In addition, Nebe and Sloane maintain an extensive catalog of known lattices online [6].

Of particular importance for dirty-paper codes are the *root* lattices,[3] Z_n, A_n, D_n, and E_n. These are defined with the following simple rules:

[2] It also implies that the double dual Λ^{**} is equal to the original lattice Λ.

[3] These are called "root" lattices for technical reasons beyond the scope of this chapter. See Conway and Sloane [84] for the details.

- Z_n is the lattice of vectors in $\mathcal{R}^n$ with integer coordinates. This is essentially the same as the lattice used in the E_LATTICE/D_LATTICE system.
- A_n is the sublattice of Z_{n+1} such that the sum of the coordinates is equal to 0. Note that this actually defines an n-dimensional lattice embedded in an $n+1$-dimensional space. A_2 is the familiar hexagonal lattice, as shown in Figure 6.6(a). This lattice will not be discussed further.
- D_n is the sublattice of Z_n such that the sum of the coordinates is equal to 0 modulo 2. This gives the so-called *checkerboard* lattice. In two dimensions, it is just an orthogonal lattice oriented at a 45° angle with the coordinate axes, as shown in Figure 6.6(b). In three dimensions, it produces the *face-centered cubic* lattice, which, by many measures, is the best three-dimensional lattice known.
- The E_n lattices do not have a simple definition that covers all possible values of n, but the E_8 lattice, which we will be using below, does have a simple definition: The lattice E_8 is the disjoint union of D_8 and $D_8 + 0.\mathbf{5}\mathbf{1}$, where $\mathbf{1}$ is the vector $(1, 1, 1, 1, 1, 1, 1, 1)$.

Based on these definitions, it is possible to construct simple and efficient quantization algorithms [84]. We describe the algorithms for all but the A_n lattices. For ease of notation, we define Q_Z as the uniform one-dimensional scalar quantizer:

$$Q_Z(x) = \lfloor x + 0.5 \rfloor. \tag{6.12}$$

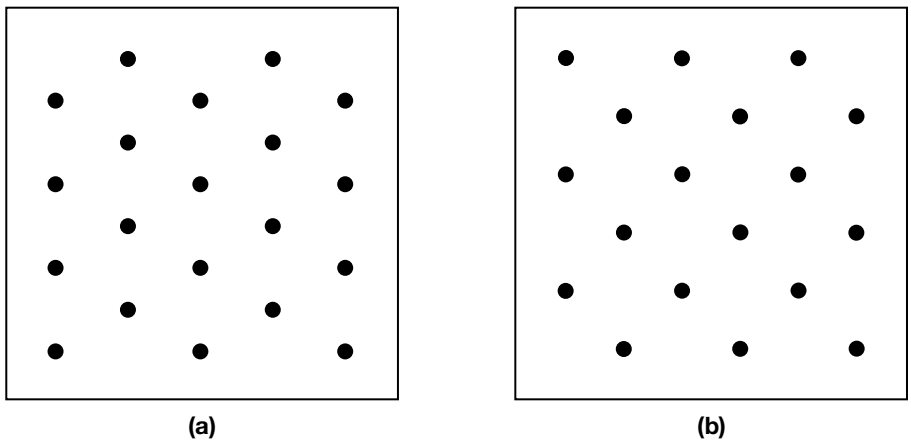


FIGURE 6.6

Arrangement of points in (a) A_2, or hexagonal, lattice, and (b) D_2, or rotated orthogonal, lattice.

Quantizing a real-valued vector, $\mathbf{x}$, to the nearest vector in Z_n is simply a matter of rounding each coordinate of $\mathbf{x}$ to the nearest integer using quantizer Q_Z. We denote this quantizer as Q_{Z_n}.

To quantize $\mathbf{x}$ to the nearest point D_n we first quantize to the nearest point in Z_n using Q_{Z_n}. If this nearest point is in D_n (i.e., the sum of the coordinates is even) we are done. If not (i.e., the sum of the coordinates is odd), then find the coordinate of $\mathbf{x}$ that was changed the most during quantization to Z_n, and round that coordinate the other way.

Rounding to the second-nearest integer like this changes the sum of the quantized coordinates from an odd number to an even number. Because the coordinate we change is the one that was furthest away from an integer to begin with, this introduces the smallest possible change in the distortion, and the resulting point is the closest one in D_n. We denote this quantizer as Q_{D_n}.

Finally, quantizing $\mathbf{x}$ to E_8 is a simple matter of quantizing to D_8 and to $D_8 + 0.5\mathbf{1}$, and choosing the quantized vector that is closest to $\mathbf{x}$.

6.5.2 Important Properties of Lattices

There are many ways to measure the quality of a lattice, and several of them are relevant to the question of whether a lattice will be good for our purposes. The most obvious of these is the quantization error, defined as the expected value of $|\mathbf{x} - Q_\Lambda(\mathbf{x})|^2$, where Q_Λ is the quantizer for lattice Λ. This directly gives us the embedding distortion we should expect when using this lattice.

A second important property of a lattice is its *sphere packing density*. Given a minimum distance between code vectors, which is determined by the amount of noise we need to survive, we would like to pack as many vectors within a region of space as possible. This will maximize the amount of information we can transmit and minimize the distortion required to transmit it. This is equivalent to imagining a fixed-size sphere around every code vector (the size of the sphere determines the robustness) and trying to pack them as tightly as possible. Figure 6.7 shows the sphere packing obtained in two dimensions with the integer lattice, Z_2, and the hexagonal lattice, A_2, illustrating that the latter packing is superior.

Recalling the proof of the Shannon theorem of Chapter 5, we know that in the limit, as the dimension n goes to infinity, we have perfect packing. In other words, for the dimension n sufficiently large, the empty space between packing spheres can, in a relative sense, be made arbitrarily small. However, for values of n much smaller than infinity, the choice of a proper lattice can have significant impact on the performance of a dirty-paper code.

The standard measure of the quality of a lattice's sphere packing is its *packing density*, which is obtained by comparing the volume of one sphere against

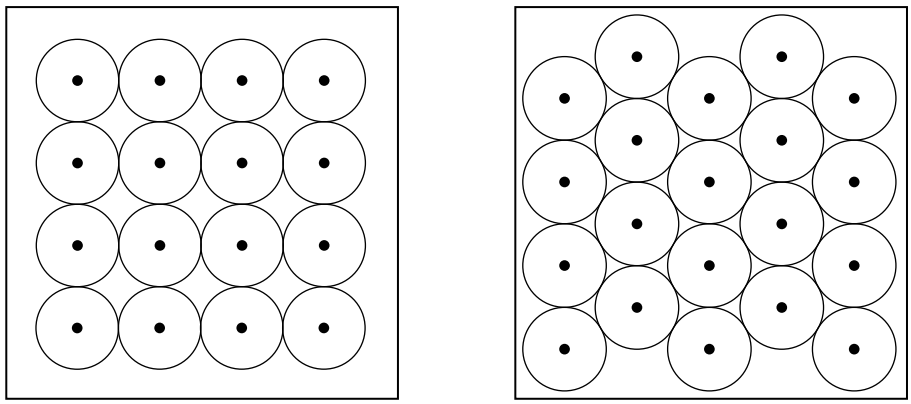


FIGURE 6.7

Sphere packing obtained with (a) orthogonal (Z_2) lattice and (b) hexagonal (A_2) lattice.

the volume of the lattice's *Voronoi cells*. A Voronoi cell around a lattice point is the set of all vectors that are nearer to that point than to any other point in the lattice. All Voronoi cells in a lattice have the same shape, so we usually only concern ourselves with the cell around the zero vector, referred to as the *basic* Voronoi cell. To measure the density of sphere packing, we find the largest sphere that can fit inside one Voronoi cell (the largest sphere we can pack into the lattice) and divide its volume by the volume of the Voronoi cell. This tells us the fraction of the total space that will be occupied by packed spheres.

A third property that is sometimes of interest is the lattice's *kissing number*. This is found by looking at one sphere in the packing and counting the number of other spheres that just touch it, or "kiss" it. For example, Figure 6.7 shows that the kissing number for Z_2 is 4, and the kissing number for A_2 is 6. Higher kissing numbers are generally considered to be better.

Not all the properties of every lattice can be calculated. However, the above values are known for many low-dimensional lattices, and this allows us to identify the best lattice for a given dimensionality in each of the senses discussed here—best quantization error, best sphere packing, and highest kissing number. In addition, for certain dimensions the theoretical best values and the corresponding lattices are known. Table 6.1 shows the best-known lattices according to each of these properties, in a number of dimensions [84].

Of course, another issue of critical importance in choosing a lattice is whether we know how to build an efficient quantizer for it. Quantizers for the root lattices discussed above are easy to implement, but it is very difficult to implement efficient quantizers for many lattices that are known to have good theoretical properties.

Table 6.1 Best-known lattices of each of the properties described above.

Dimension	1	2	3	4	5	6	7	8
Best quantizer	Z	A_2	A_3	D_4	D_5	E_6	E_7	E_8
Densest packing	Z	A_2	A_3	D_4	D_5	E_6	E_7	E_8
Kissing distance	Z	A_2	D_3	D_4	D_5	E_6	E_7	E_8

6.5.3 Constructing a Dirty-Paper Code from E_8

Once we have chosen a base lattice, Λ, we must decide which vectors to use for our message-dependent dither, $C(m)$. Intuitively, it is clear that we need to maximize the distance between any point that represents one message and the closest point that represents a different message. That is, we want to choose $C(m)$ such that

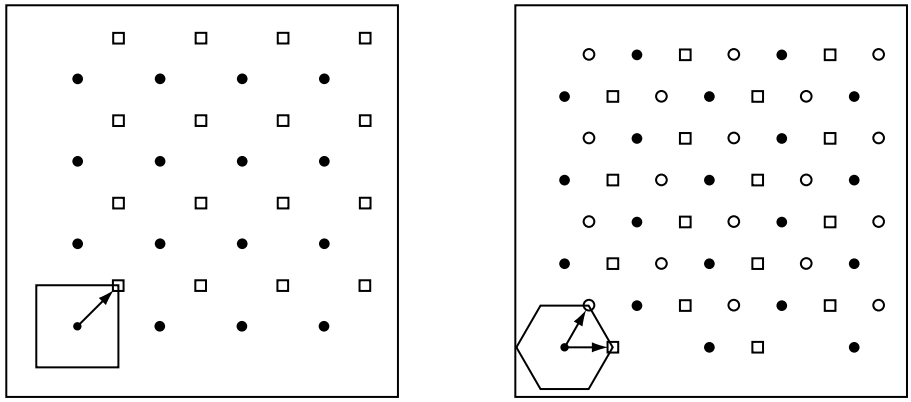
$$D_{ij} = \min_{\mathbf{x}\in\Lambda+C(m_i),\mathbf{y}\in\Lambda+C(m_j)} |\mathbf{x}-\mathbf{y}| \tag{6.13}$$

is maximized (when $i \neq j$). By maximizing this distance, we maximize the amount of noise required to move from the detection region around one code vector into the detection region for another. More formal justification for this intuition can be found in Moulin and O'Sullivan [302].

Without loss of generality, we can begin by choosing $C(m_0)$ to be the zero vector, so that $\Lambda + C(m_0) = \Lambda$. For the next message, we can choose a vector $C(m_1)$ that is as far from the zero vector as possible, without being closer to any other vector in Λ. In other words, $C(m_1)$ should be the largest-magnitude vector in the basic Voronoi cell of the lattice. Offsetting the lattice by this vector will result in a set of points that are as far away from the original lattice as we can get. There may be additional vectors in the Voronoi cell that have the same magnitude as $C(m_1)$, and these can be used for additional messages.

The vectors we are choosing here are known as the *deep holes* of the lattice. Figure 6.8 shows the locations of the deep holes in the two-dimensional orthogonal (Z_2) and hexagonal (A_2) lattices. The orthogonal lattice has just one deep hole, so it can make a good code for only two distinct messages. The hexagonal lattice has two deep holes, so it can make a good code for three messages.

It is pointed out in Moulin and O'Sullivan [302] that, for small numbers of messages, we should choose a lattice that has the right number of deep holes and no more. Though the orthogonal lattice is clearly inferior to the hexagonal one in terms of quantization error, sphere packing, and kissing number, it is actually *better* for a two-message (1-bit) code. This is because the distance to the orthogonal lattice's single deep hole is $\sqrt{2} \approx 1.41$, whereas the distance to each of the hexagonal lattice's two deep holes is $(2/3)\cos(30^\circ) \approx 0.58$.



(a) (b)

FIGURE 6.8

Arrangement of deep holes in (a) orthogonal (Z_2) lattice and (b) hexagonal (A_2) lattice. In the lower left of each diagram is an illustration of one Voronoi cell, with the vectors that yield the deep holes.

Of course, the embedding distortion introduced by the hexagonal lattice is smaller than that of the orthogonal, so we can use a coarser hexagonal lattice, but the difference is not enough to overcome the distance to the deep holes. The real advantage that the hexagonal lattice gives us is the availability of three good code vectors. If we don't use them all, we are better off with the simpler lattice.

INVESTIGATION

Watermarking with the E_8 Lattice

We now provide an example watermarking system, which, like the system presented in Zhanq and Bosten [467], uses the E_8 lattice in place of the orthogonal lattice of E_LATTICE/D_LATTICE. There are several reasons to choose the E_8 lattice here. The first two appear in the text above: (1) the E_8 lattice has the best properties among known eight-dimensional lattices, and (2) there exists a simple and efficient quantization algorithm for it.

The third reason for choosing the E_8 lattice is that we can find 15 deep holes. As it turns out, the union of the E_8 lattice and its shifted versions to the 15 deep holes form a new lattice, Γ. This lattice, Γ, has E_8 as a sublattice with 16 different cosets. The pair (Λ, Γ) can therefore be used as a dirty-paper lattice for 16 different messages.

System 10: E_E_8LATTICE/D_E_8LATTICE

The deep holes can be computed as linear combinations of the following four row vectors [467]:

$$P = \frac{1}{4}\begin{bmatrix} 0 & 0 & 0 & -2 & -2 & -2 & 2 & 0 \\ 1 & 1 & -3 & -1 & 1 & -1 & -1 & -1 \\ 0 & 2 & 0 & -2 & 0 & 0 & -2 & -2 \\ 0 & 0 & 2 & 2 & 0 & 2 & 0 & -2 \end{bmatrix}. \tag{6.14}$$

We now have enough to make a DM watermarking system that embeds 4 bits of information in an eight-dimensional vector. To embed a larger number of bits into a higher-dimensional vector, we merely divide the vector into groups of eight dimensions and embed 4 bits into each.

There remains the matter of choosing the dimensions in which we will embed. We wish to compare the system against E_LATTICE/D_LATTICE, so we need to embed the same number of bits into each image (345 bits, trellis coded to 1,380 bits, embedded in each 240 × 368 image). As that system embeds 1 coded bit into each projected dimension, and the E_E_8LATTICE/D_E_8LATTICE system embeds half a coded bit into each projected dimension (4 bits into every eight dimensions), we must project into twice as many dimensions. We will do this by correlating each 8 × 8 block of the image with two reference marks, $\mathbf{w_{r0}}$ and $\mathbf{w_{r1}}$, instead of just one.

Thus, the E_E_8LATTICE embedding algorithm proceeds in the following steps:

1. Encode the message into a sequence of coded bits, $m_c[1], m_c[2], \ldots, m_c[1380]$.
2. Divide the cover image, $\mathbf{c_0}$, into 1,380 8 × 8 blocks.
3. Correlate each block, $\mathbf{c}_i$, with two reference marks, $\mathbf{w_{r0}}$ and $\mathbf{w_{r1}}$, to obtain a vector, $\mathbf{v_0}$, of length 2,760.
4. Using distortion compensation with parameters (α, β) (see Section 6.4.2) and an E_8 lattice, Λ, apply DM to every eight dimensions in $\mathbf{v_0}$ to embed 4 bits of m_c.
5. Modify each block, $\mathbf{c}_i$, of the image to embed the watermark.

The D_E_8LATTICE detector performs the following steps:

1. Project the image, $\mathbf{c_r}$, into a 2,760-dimensional vector, $\mathbf{v_r}$, as in Steps 1–3 of the embedder.
2. For every eight dimensions in $\mathbf{v_r}$, try quantizing with the quantizers for all 16 possible 4-bit messages. The quantizer that results in the smallest distortion indicates these 4 bits of the trellis-coded message.
3. Apply the trellis decoder to obtain the embedded message.

Experiments

In this experiment we chose $\alpha = 0.7538$ and used a scaled version of E_8 such that the average per-pixel distortion was equal to 2. We embedded a random

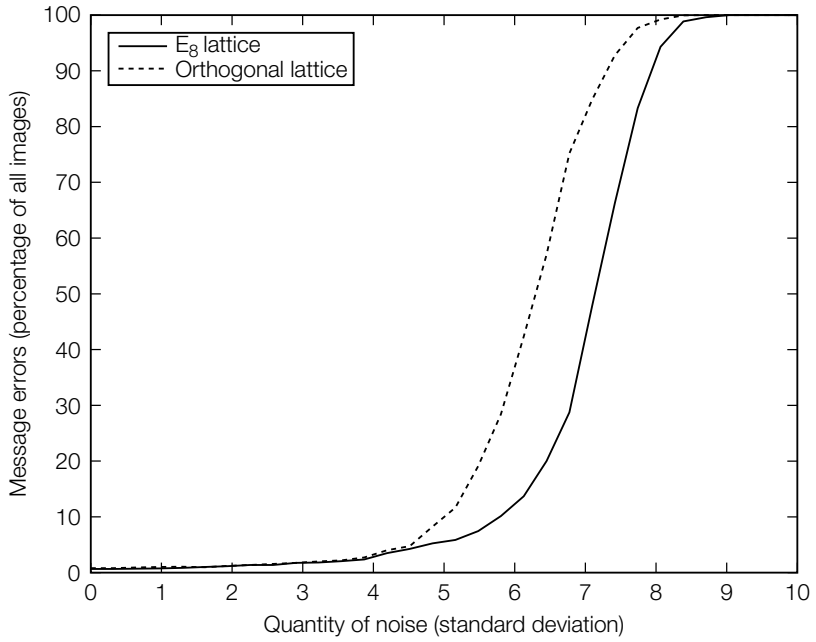


FIGURE 6.9

Effect of additive white Gaussian noise on E_E_8LATTICE/D_E_8LATTICE watermarking system. Results for the E_LATTICE/D_LATTICE system are shown with a dotted line for comparison.

message in each of 2,000 images (different message in each image) with E_LATTICE, and ran D_LATTICE to see if it was correctly embedded. This yielded an embedding effectiveness of 99.35%.

We tested for robustness against additive noise by running the same test we applied to E_LATTICE/D_LATTICE. The results are shown in Figure 6.9. The results from E_LATTICE/D_LATTICE are also shown in this figure with a dotted line. Clearly, the watermarks embedded using the E_8 lattice outperformed those embedded with the orthogonal lattice. The median improvement in the magnitude of noise that watermarks could survive was about 15%.

6.6 MAKING LATTICE CODES SURVIVE VALUMETRIC SCALING

Lattice codes are simple to implement and give very good robustness against AWGN distortion. In fact, with the right types of lattices, they can achieve the full capacity of Costa's dirty-paper channel [299]. For these reasons, they have become the method of choice in dirty-paper coding research for many applications. However, lattice codes have a serious weakness that must be

addressed before they can produce robust watermarks: If you multiply the samples of a watermarked Work by a value even mildly different from 1, a lattice-based watermark will not survive.

Scaling the sample values in a Work is termed *valumetric scaling* (in Chapter 9 we use the term *valumetric* to distinguish this from *geometric scaling*, which refers to resampling the Work with a different sampling frequency, i.e., changing its size). In images, valumetric scaling corresponds to increasing or decreasing the contrast. In audio, it increases or decreases the volume. As such, it is a very benign distortion—most observers would not consider it a distortion at all, in that it usually doesn't change the perceived quality of the Work.

That valumetric distortion will destroy a lattice-based watermark can be seen in Figure 6.10. This illustrates what happens to such a mark when it is scaled by a factor of 0.8. Even though this is a relatively small change, the absolute change is large enough to move the mark into the detection region of a different message. Experiments with the E_LATTICE/D_LATTICE system bear this out, as shown in Figure 6.11. Increasing contrast by a factor of just 1.16, or decreasing by 0.85, resulted in message error rates above 20%.

In this section, we examine the main approaches that have been proposed for solving this problem.

6.6.1 Scale-Invariant Marking Spaces

Most authors writing about lattice-based watermarks assume that they are working in a marking space where all important distortions are manifested, at worst,

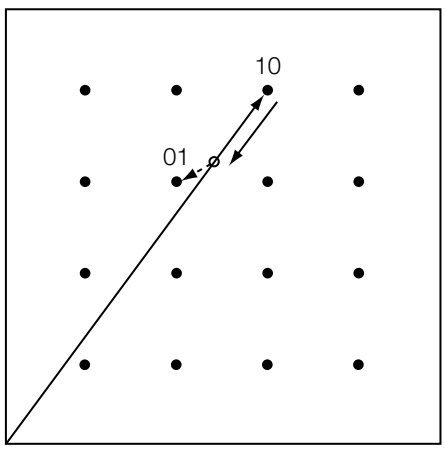


FIGURE 6.10

Effect of valumetric scaling on a lattice-coded message. The long, solid arrow shows a vector quantized to encode the message 10. The short, solid arrow and open circle show the effect of scaling this by a factor of 0.8. The dashed arrow indicates that the resulting point will be decoded as the message 01.

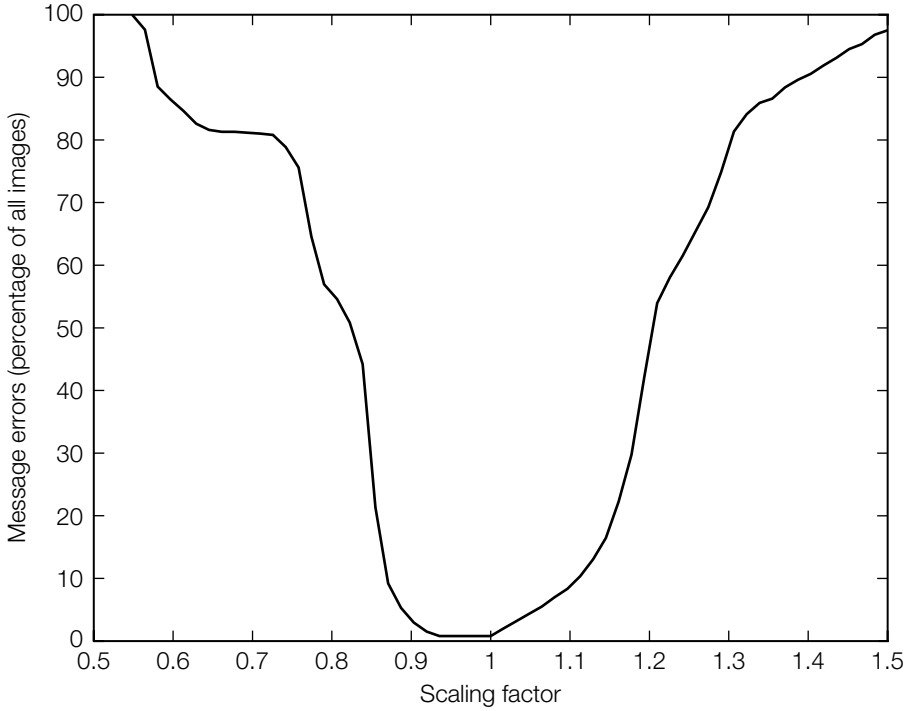


FIGURE 6.11

Effect of valumetric scaling on the E_LATTICE/D_LATTICE system.

as additive noise. Valumetric scaling is an important distortion, so such a space would have to be more or less scale invariant. That is, if a Work is scaled by even a large amount, its mapping into marking space should not change much.

Ideally, the marking space should be a full-fledged *perceptual space*, meaning that Euclidian distance in marking space is monotonically related to perceptual distance in an observer's judgment (see Chapter 8). Two works that are far apart perceptually should be far apart in marking space. Works that are perceptually similar should be close together in marking space. As valumetric scaling has little perceptual importance, it would not result in big changes in a perceptual space.

Much theoretical analysis of watermarking assumes that it is applied in a perceptual marking space. Unfortunately, such a space is very difficult to create in practice. Simple distortions such as shifting an image or rotating it slightly can cause huge differences in its original, media space vector, but have essentially no perceptual impact. Even worse, slightly moving a single object *within* an image has little perceptual impact, but again can change the pixel values a great deal. Mapping images into a truly perceptual space, then, could require such difficult steps as segmenting them into objects and finding some sort of canonical registration.

In practice, a more tractable approach is to define marking spaces that are only invariant to specific (and simple) distortions that we know to be a problem. Thus, in our case here, we want to find some space that is reasonably invariant to valumetric scaling.

One could embed watermarks into the phases of complex Fourier coefficients for images. Valumetric scaling does not change phase information (it changes only the coefficients' magnitudes), so watermarks embedded here should be unaffected by such distortions.

To embed watermarks in Fourier phases, we need to employ a lattice that accounts for the inherent circularity of a phase. That is, we need to account for the fact that a phase of 359° is close to a phase of 0°. This is a simple matter of choosing a quantization step size (β in the example watermarking systems above) that divides the circle into an integral number of segments, and then applying quantization with mod-360° arithmetic. To improve fidelity, we can use different quantization step sizes for coefficients of different frequencies, accounting for their differing perceptual importance.

Note that phase watermarking is not appropriate for audio. Although a phase is critically important in image perception (see Chapter 8), it is nearly irrelevant in audio perception. This means that many phase-changing audio processes can be applied without changing the quality of the signal, and a watermark embedded in the phase of an audio signal will be susceptible to these distortions. Audio therefore requires a different scale-invariant domain.

6.6.2 Rational Dither Modulation

A second approach to achieving scale invariance in lattice watermarks is to dynamically adjust the quantization step size for each dimension according to information available elsewhere in the signal. If this is done in such a way that linear scaling of the latter information results in linear scaling of the quantization step sizes, then the watermark should be robust against such changes. This is the approach taken in *rational dither modulation* (RDM) [327].

In RDM, we assume a sequential watermarking process and adapt the quantizer in each step based on features of the previously marked components. When the embedder is modifying a given vector in marking space, $\mathbf{v}$, into a watermarked vector, $\mathbf{v}_m$, then for each successive element of the vector, $\mathbf{v}[i > k]$, it computes a function, $g(\mathbf{v}_m[i-k \ldots i-1])$, of the preceding k watermarked elements. It uses this value as the scaling factor, β, in a basic lattice embedding algorithm like E_LATTICE. The function, g, should be homogeneous as defined by

$$g(s\mathbf{x}) \approx sg(\mathbf{x}) \tag{6.15}$$

for any scalar s.

When the detector is decoding a watermark in a received marking space vector, $\mathbf{v}_r$, it computes each value of $g(\mathbf{v}_r[i-k \ldots i-1])$ as in the encoder, and uses it to decode the bit value embedded in $\mathbf{v}_r[i]$. Note that, if the work has

not been distorted, then $\mathbf{v}_r = \mathbf{v}_m$, and the values of g used during decoding will be identical to those used during embedding. If the work has been subjected to valumetric scaling, then $\mathbf{v}_r = s\mathbf{v}_m$, so $g(\mathbf{v}_r[i-k \dots i-1]) \approx sg(\mathbf{v}_m[i-k \dots i-1])$, and the quantization step size for each $\mathbf{v}_r[i]$ will be approximately scaled by s, which should yield the correct results.

An intriguing definition of g for image watermarking was suggested in Li and Cox [256]. Here, g is a variant of Watson's perceptual model [438] (this model is described in detail in Chapter 8), modified to satisfy the scaling property in Equation 6.15 and to employ only the k preceding DCT coefficients when computing the perceptual slack for coefficient i. This means that $g(\mathbf{v}_m[i-k \dots i-1])$ will tend to be large when the values of $\mathbf{v}_m[i-k \dots i-1]$ indicate that $\mathbf{v}[i]$ is perceptually insignificant, and small when they indicate that it is perceptually important. Thus, at the same time that we solve the problem of making a lattice code robust against valumetric scaling, we also obtain some perceptual adaptation. Other related work includes [257, 314].

6.6.3 Inverting Valumetric Scaling

Another approach to the problem of valumetric scaling is to try to estimate and invert any valumetric scaling that was applied.

This approach was first suggested by Eggers *et al.* [129]. Their initial formulation involved embedding a pilot signal, separate from the message-carrying portion of the watermark, that the detector could use to determine any scaling that had been applied. Later it was noted that, because the watermarks are embedded with a known lattice, the message-carrying portion could itself be used for this purpose [254, 371]. The basic observation is that any lattice is defined by a finite set of basis vectors. Therefore, given sufficiently many watermarked components with sufficiently many random messages, the unknown lattice can be discovered.

Let $\mathcal{P}$ be the set of points that need to be matched against a scaled version $s\Lambda$ of a lattice. A first approach to find the scaled lattice simply does a search over the expected range of the scaling parameter s. For each s, the set $\mathcal{P}$ is quantized to $s\Lambda$ and the MSE between $\mathcal{P}$ and $Q_{s\Lambda}(\mathcal{P})$ is measured. The best estimation of s is then found as the value of s that minimizes the error.

A second approach exploits the symmetry of a lattice $s\Lambda$ by noting that for any $a \in s\Lambda$, the sets $\mathcal{P}$ and $a + \mathcal{P}$ are expected to have a high correlation. This symmetry property is best analyzed in the frequency domain using a Fourier transform and has been explored by Sidorov [372]. Note that this second approach is related to X-ray crystallography, which exploits symmetry properties of crystals by means of diffraction patterns [9].

6.7 DIRTY-PAPER TRELLIS CODES

Dirty-paper trellis codes were originally designed to address the sensitivity to valumetric scaling by placing the code words on the surface of a multidimensional

sphere rather than on a lattice [288, 291]. These *spherical codes* all have the same energy (i.e., are equi-energetic) and are unaffected by amplitude scaling of the signal.

The main design goals for spherical codes are that the cosets are well separated to provide robustness to noise, and the code words within each coset are uniformly distributed over the sphere in order to minimize the distortion to the content.[4] And, of course, there should be computationally efficient algorithms for encoding and decoding.

The design of dirty-paper spherical code books is difficult so let us first look at how spherical codes are implemented using a dirty-paper trellis. The concept of a trellis was introduced in Chapter 4 for error correction purposes and discussed in the Investigation, "Watermarking with an Orthogonal Lattice Code," earlier in this chapter. This traditional form of trellis is reproduced in Figure 6.3.

The reader is reminded that each arc of the trellis is labeled with a sequence of bits and each path through the trellis encodes a unique message. To construct a dirty-paper trellis, we fix an orthogonal decomposition of our space of signals. That is, each signal $\mathbf{x}$ can uniquely be written as a sum

$$\mathbf{x} = \sum_{i=1}^{m} x[i]. \tag{6.16}$$

Each $x[i]$ is represented by a (typically short) real-valued vector. Each arc in the i^{th} stage of a dirty paper trellis is now labeled with a real-valued vector corresponding to the i^{th} phase in the chosen orthogonal decomposition. The vectors corresponding to all possible paths through the trellis comprise the set of points to which we can quantize. The cost function associated with an arc can be any additive nonnegative function D, that is, any function D that satisfies

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} D(\mathbf{x}[i], \mathbf{y}[i]). \tag{6.17}$$

Typically D will be defined as an L_p norm, with the most common value for p being equal to 2 (i.e., the squared error norm).

The next modification is to alter the cost associated with each arc. In Chapter 4 this cost was the Hamming distance. However, for a dirty-paper trellis, the Hamming distance is replaced with any distance function between two real-valued vectors, provided that function can be computed additively along the elements of the vectors:

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i} d(\mathbf{x}[i], \mathbf{y}[i]). \tag{6.18}$$

where $D(\mathbf{x}, \mathbf{y})$ is the distance between vectors $\mathbf{x}$ and $\mathbf{y}$, and $d(x, y)$ is a symmetric function of two scalars, x and y.

[4] Note that in this respect, trellis codes are in the spirit of the ideal Costa scheme, which is also characterized by uniformly distributed code words on a high-dimensional sphere.

For example, if we are embedding 1 bit in each 8 × 8 block of pixel values, each arc may have an 8 × 8 random vector, $\boldsymbol{x}$, associated with it, and $\boldsymbol{y}$ is the corresponding 8 × 8 block of pixel values. The cost associated with the arc might then be the sum of the squared differences between the vector elements. Instead of measures of distance, we can also use measures of similarity, such as linear correlation.

At this point, we still do not have a dirty-paper code. To create this we modify the trellis structure so that more than two arcs emanate from each state, as depicted in Figure 6.12. To embed a binary message, arcs are also assigned binary labels, represented by the bold and nonbold arcs of Figure 6.12. Thus, a path through the trellis represents a binary sequence of bits, and multiple paths represent the *same* sequence. The multiple paths that encode the same message represent one coset. Thus, we now have our dirty-paper code.

Efficient decoding of the dirty-paper trellis code can be accomplished by applying Viterbi's algorithm to the dirty-paper trellis. The Viterbi algorithm simply finds the closest code word to the watermarked Work, where the closest is determined by the distance function associated with each arc. This search is performed over the union of all the cosets.

However, for the watermark embedder it is necessary to find the closest code word in the coset that encodes the desired message, not in the union

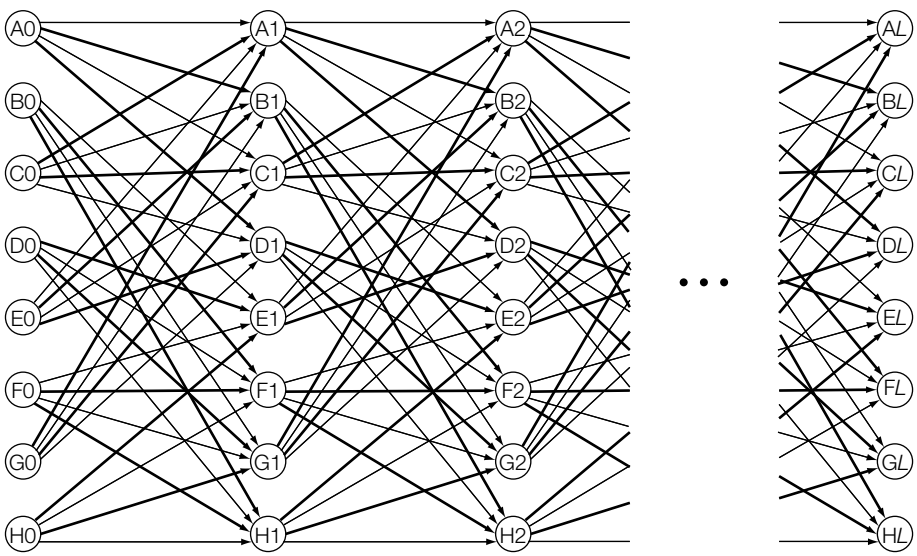


FIGURE 6.12

A dirty-paper trellis in which more than two arcs emanate from each state. The bold arcs represent a message bit, "1," and nonbold arcs represent a message bit, "0."

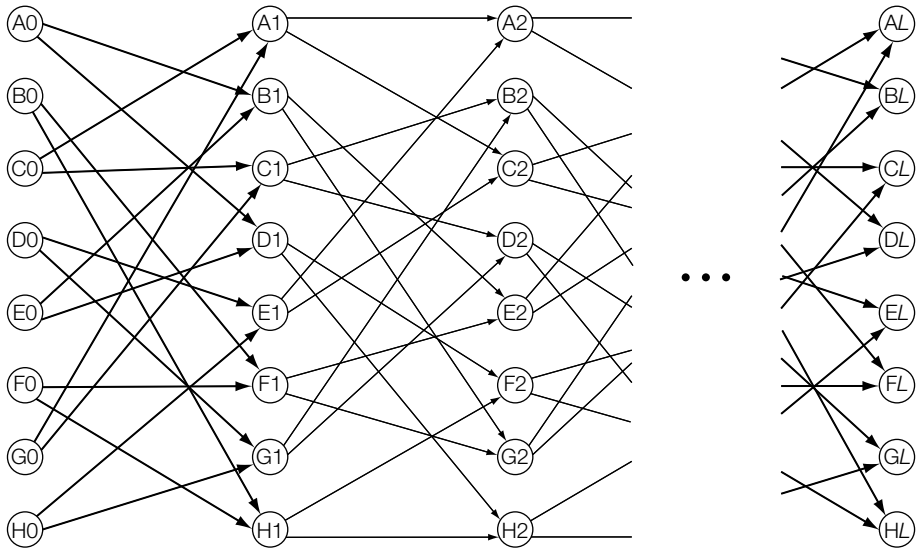


FIGURE 6.13

A modified dirty-paper trellis in which only arcs that represent the desired message have been retained. Note that all paths through the trellis encode the same message.

of all cosets. To accomplish this, we first modify the full dirty-paper trellis, retaining only those arcs that encode the desired message. A modified dirty-paper trellis is illustrated in Figure 6.13. All paths through this modified trellis encode the same message (i.e., the modified trellis represents all the code words in a single coset). To find the best code word in the coset, we again use Viterbi's algorithm, but on the modified dirty-paper trellis, and with the original cover Work as input.[5]

At this point, we have efficient means to encode and decode the dirty-paper trellis codes. However, this does not guarantee that the spherical codes generated by the trellis will have cosets that are well separated, or that code words within each coset will be uniformly distributed over the sphere. This remains a problem. Originally, the vectors associated with the arcs of the trellis were randomly generated. More recently, there has been interest in more principled designs, [433], based on trellis-coded modulation [333, 415]. And Abrardo *et al.* [13] reports work on a combination of rational dither modulation and dirty-paper trellis coding.

[5] Note again, this is very close in spirit to the ideal Costa scheme. Also note, that lattice coding follows the same procedure: The embedder quantizes to the coset code words, whereas the decoder quantizes to the union of all cosets.

6.8 SUMMARY

The ideal "Writing on Dirty Paper" method relies on signals of infinite length and well-behaved distortion models. They therefore cannot easily be applied in practical settings and approximations have to be found. In this chapter we considered a number of approaches to apply Costa's method in practical settings.

- Costa's results need to be adapted to apply to finite dimensional real contexts.
- The choice of the right lattices and quantizers, the appropriate distortion compensation strategy, and random dither are important elements in building high-performance practical systems.
- Classical root lattices are suitable for quantization watermarking. In particular, the E_8 lattice has high-performance parameters and can be easily implemented.
- Distortion compensation in the finite case may exploit an additional degree of freedom for scaling the global lattice.
- Random dither may be used to preserve signal statistics and add more degrees of freedom to the quantizers. The latter is important for making lattice quantizers more secure.
- Many realistic distortions with minimal perceptual impact are difficult to model as MSE types of distortions. Two important examples are given by geometric and valumetric scaling distortions.
- Quantization-type watermarking methods are particularly sensitive to valumetric scaling distortions.
- Valumetric scaling distortions can be resolved by choosing an invariant marking space, adaptive quantizing, or inverting the scaling parameter.
- Dirty-paper trellis codes are invariant to valumetric distortion due to their use of spherical codes.

CHAPTER 7

Analyzing Errors

Errors are inevitable in even the best-designed watermarking systems. In this chapter, we discuss three types of errors: false positive errors, false negative errors, and message errors. A *false positive error* occurs when the detector incorrectly indicates that a watermark is present. Conversely, a *false negative error* occurs when a detector incorrectly indicates the absence of a watermark. A *message error* occurs when a watermark detector incorrectly decodes a message.

The designer of a watermarking system must determine what error rates are acceptable during the specification phase of the design. It is therefore necessary to develop models for the errors of interest. The purpose of these models is twofold. First, a model allows us to select a detection threshold to meet the specifications. Second, experimental verification of the model allows us to be confident that the specified error rates will not be exceeded.

The severity of these errors depends on the application. For example, in a broadcast monitoring application to confirm that advertisements are aired, a false negative is very serious, in that it leads to the erroneous conclusion that an advertisement was not broadcast. This, in turn, may have serious adverse repercussions on the relationship between the broadcaster and the advertiser. In contrast, false positives lead to the conclusion that the advertisement is being aired more frequently than expected. This situation is also erroneous, but it does not result in mistrust or litigation between the broadcaster and the advertiser.

Message errors are considered in Section 7.1. (The reader is also directed to Chapter 4). The false positive probability is examined in detail in Section 7.2. A simple Gaussian error model is described and the concepts of random-watermark and random-Work false positives are introduced. The section concludes with an Investigation that highlights the differences between these two forms of false positives and the importance of accurate modeling. The false negative probability is highly affected by the distortions the Work undergoes between the times of embedding and detection. Section 7.3 discusses this issue and the relationship between false negative errors and security, robustness, and

effectiveness. It should be noted that whereas false positives depend only on the detection algorithm, false negatives also depend on the embedding algorithm. We have seen in Chapter 5 that different embedding algorithms can have very different robustness characteristics.

False positive and false negative errors are coupled. For example, it is easy to guarantee no missed detections (i.e., no false negatives) if we set our threshold to the smallest value possible. However, the number of false positive errors then becomes overwhelming. Section 7.4 introduces the concept of a receiver operating charactertisic (ROC) curve as a method of presenting the tradeoff between the false positive and false negative rates.

It is common to assume that the additive noise that distorts a watermark is white (i.e., the elements of the noise vector are uncorrelated with one another). However, there are many cases for which this is not true. The Investigation of Section 7.2 includes such a case. When this assumption is false, the error model can perform very poorly. To solve this problem, Section 7.5 describes the use of a whitening filter commonly used to decorrelate noise prior to detection. It is well known that when a signal is corrupted by additive white Gaussian noise, linear correlation is the optimum detection statistic. Thus, whitening improves detection performance by transforming a signal corrupted by nonwhite noise into one in which the noise is white and optimum detection can be achieved.

In Section 7.6, we utilize the concepts outlined in Sections 7.2 through 7.5 to measure and model the errors present in some example algorithms based on using the normalized correlation statistic for detection.

7.1 MESSAGE ERRORS

A message error occurs when a watermark decoder incorrectly decodes a message. When direct message coding is used, the detector mistakenly decodes one message when, in fact, another message was embedded. Similarly, for multisymbol messages, the detector will erroneously decode one or more symbols. When a binary alphabet is used, these errors are *bit errors,* and the *bit error rate,* or BER, is a measure of the frequency of bit errors.

Bit, symbol, or message errors occur when noise distorts the embedded signal such that the signal is moved from one detection region to another. This is why maximizing robustness to distortion requires maximizing the separation between codes. Chapter 4 discussed a number of ways in which this can be achieved.

One common method of protecting multibit or multisymbol messages is to use some form of error detection and correction code. There are many such codes [163], and the choice of a particular code will depend on the types of errors expected in the watermark application and the computational constraints of the design. For example, in an image watermarking application, cropping may be a common distortion. If the symbols of the message are encoded using spatial multiplexing in which each successive bit is embedded in the next spatial region, cropping will result in *burst errors*. That is, a sequence of successive

bits will either be deleted or corrupted. In this case, it is appropriate to choose an error correction code that is robust to burst errors. Cyclic codes (such as the Reed-Solomon code) have good burst error properties. Another solution to the problem of burst errors due to cropping is to randomize the spatial location of each encoded bit. In this way, cropping no longer manifests itself as a burst error but as a random error. In this case, block codes may be most suitable, such as parity-check codes.

An error detection and correction code will specify the maximum number of bit errors it can detect. If the actual number of bit errors exceeds this maximum, uncorrected bit errors will occur. The seriousness of these errors will depend strongly on the application. In some applications, certain message errors are more serious than others. For example, in a copy control application we might have a copy-once message, indicating that a recorder may make a first-generation copy of a Work but may not make a copy of the copy (see Section 2.3.9 of Chapter 2). Confusing this message occasionally with copy-freely might not be a serious problem, in that it might allow second-generation copies of only portions of Works (e.g., individual movie scenes or brief segments of songs). On the other hand, confusing it with never-copy, even very rarely, could be a severe problem, in that it would prevent users from making first-generation copies they are entitled to make and would be perceived as equipment malfunctions.

When messages are encoded with sequences of symbols, critical pairs of messages, such as copy-once and never-copy in the previous example, should differ in as many symbols as possible to reduce the chance that they will be confused. It is also possible to intentionally design watermarking systems in which symbols are embedded with varying levels of reliability. For example, [452] describes a system in which the bits of a message are divided into two groups. A small group of bits is embedded with a high degree of robustness, and a larger group of bits is embedded with less robustness. In this system, critical information can be encoded in the robust bits, whereas the other bits are used to encode less important information.

Bit error rates can be modeled [163]. This requires a characterization of the transmission channel. For example, for binary symbols we might ask whether the channel is symmetric. That is, is the probability of detecting a 0 when a 1 was embedded the same as the probability of detecting a 1 when a 0 was embedded? Such a channel is referred to as a *binary symmetric channel*. Alternatively, the channel might be a *binary erasure channel,* in which the channel output has not just two values but a third that indicates indecision. Is the noise additive, white Gaussian? What is the probability of a single bit error? Are the errors independent or are burst errors common? Such issues have received considerable attention from the communications community, and the reader should consult these sources for further information. In addition, [80, 188, 189, 191, 244] contain a number of investigations of bit error rates for watermarking applications. The following Investigation describes a simple Gaussian model for a bit error rate that is sometimes appropriate.

INVESTIGATION

A Gaussian Model for Bit Error Rate

In Chapter 4 we described a watermarking algorithm for embedding 8 bits of data in an image. This system, E_SIMPLE_8/D_SIMPLE_8 (System 4), represents each of the 8 bits by an orthogonal, white noise pattern that is either added or subtracted depending on whether the corresponding bit is a 1 or a 0. The message mark, $\mathbf{w}_m$, is a combination of these eight patterns that is then scaled by $1/\sqrt{8}$ to have unit variance. The watermarked work, $\mathbf{c_w}$, is then given by

$$\mathbf{c_w} = \mathbf{c_o} + \alpha \mathbf{w}_m, \tag{7.1}$$

where α is an arbitrary constant. Each bit is therefore embedded with a strength of $\alpha/\sqrt{8}$.

To detect bit *i*, we linearly correlate the watermarked Work with the corresponding reference pattern, $\mathbf{w_{r}}_i$. If the correlator output is greater than zero, a 1 bit is present; otherwise, a 0 bit is present.

We assume that the detector output for each bit is Gaussian distributed. This assumption is valid for watermark patterns that are white. However, we will see in the next section that this assumption is invalid when the watermark patterns are not white.

For each bit, the mean value of the linear correlation, μ_{lc}, is

$$\mu_{\mathrm{lc}} = \frac{\alpha}{\sqrt{2}}, \tag{7.2}$$

and the variance, σ^2_{lc}, is

$$\sigma^2_{\mathrm{lc}} = \sigma^2_{\mathbf{w_r}i}\left(\sigma^2_{\mathbf{c_o}} + \sigma^2_n\right), \tag{7.3}$$

where the variance of each reference pattern, $\sigma^2_{\mathbf{w_r}i} = 1$, and $\sigma^2_{\mathbf{c_o}}$ and σ^2_n are the variances of the cover Work and the channel noise, respectively. The probability of a bit error is then

$$p = \int_{-\infty}^{0} \frac{1}{\sqrt{2\pi}\sigma_{\mathrm{lc}}} e^{-\frac{\left(x-\mu_{\mathrm{lc}}\right)^2}{2\sigma^2_{\mathrm{lc}}}} dx \tag{7.4}$$

$$= \int_{-\infty}^{-\mu_{\mathrm{lc}}} \frac{1}{\sqrt{2\pi}\sigma_{\mathrm{lc}}} e^{-\frac{x^2}{2\sigma^2_{\mathrm{lc}}}} dx \tag{7.5}$$

$$= \int_{\mu_{\mathrm{lc}}}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_{\mathrm{lc}}} e^{-\frac{x^2}{2\sigma^2_{\mathrm{lc}}}} dx \tag{7.6}$$

$$= \mathrm{erfc}\left(\frac{\mu_{\mathrm{lc}}}{\sigma_{\mathrm{lc}}}\right), \tag{7.7}$$

where

$$\text{erfc}(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-\frac{t^2}{2}}\, dt. \tag{7.8}$$

Experiment

To test this model, we watermarked 2,000 images with 10 random 8-bit messages and $\alpha = 2$, resulting in 20,000 watermarked images. The mean value of the linear correlation is given by

$$\mu_{lc} = \frac{2}{\sqrt{8}} = \frac{1}{\sqrt{2}}, \tag{7.9}$$

and the measured standard deviation of pixel values in the images, σ_{c_o}, was found to be 59. We then added white, Gaussian noise with eight different standard deviations, $\sigma_n = 0, 20, 40 \ldots 140$. For each noise value, we detected the 20,000 8-bit messages and determined the number of bit errors. The measured and predicted error rates are plotted in Figure 7.1.

There is good agreement between the measured and predicted rates. The deviation at low bit error rates is due to the fact that at low noise values the image noise dominates, and this noise is not Gaussian.

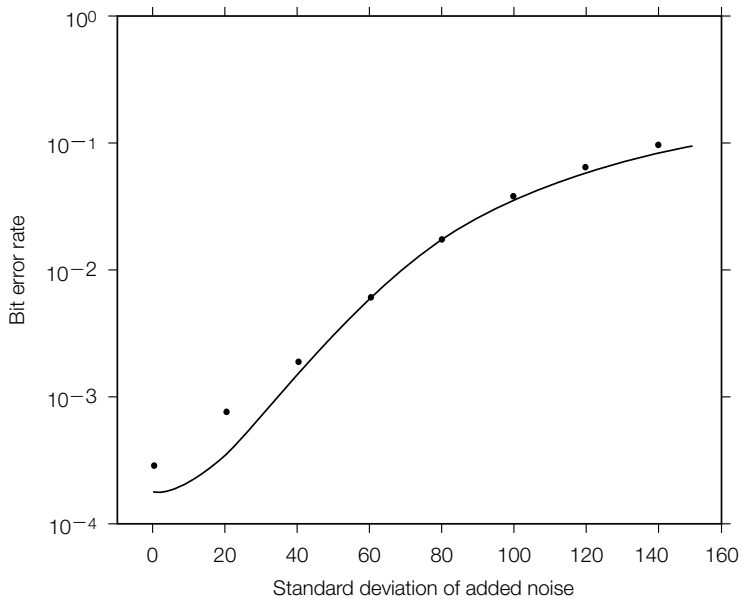


FIGURE 7.1

Measured and predicted bit error rates for E_SIMPLE_8/D_SIMPLE_8 (System 4).

7.2 FALSE POSITIVE ERRORS

A false positive occurs when a watermark detector indicates the presence of a watermark in an unwatermarked Work. The *false positive probability* is the likelihood of such an occurrence, and the *false positive rate* measures the frequency of false positives. A false positive probability of 10^{-4} indicates that we expect, on average, one false positive for every 10,000 detection attempts.

Figure 7.2 illustrates how and why false positive errors can occur. The left-hand curve represents the frequency of occurrence of each possible value that can be output from the watermark detector when no watermark is actually present. Similarly, the curve to the right represents the frequency of detector output values when a watermark is present. The vertical line represents the decision boundary, τ: If the detector output value is less than τ, the watermark is declared absent; otherwise, the watermark is declared present. False positive errors are possible because there is a finite chance the detector will output a value greater than or equal to τ when no watermark is present.

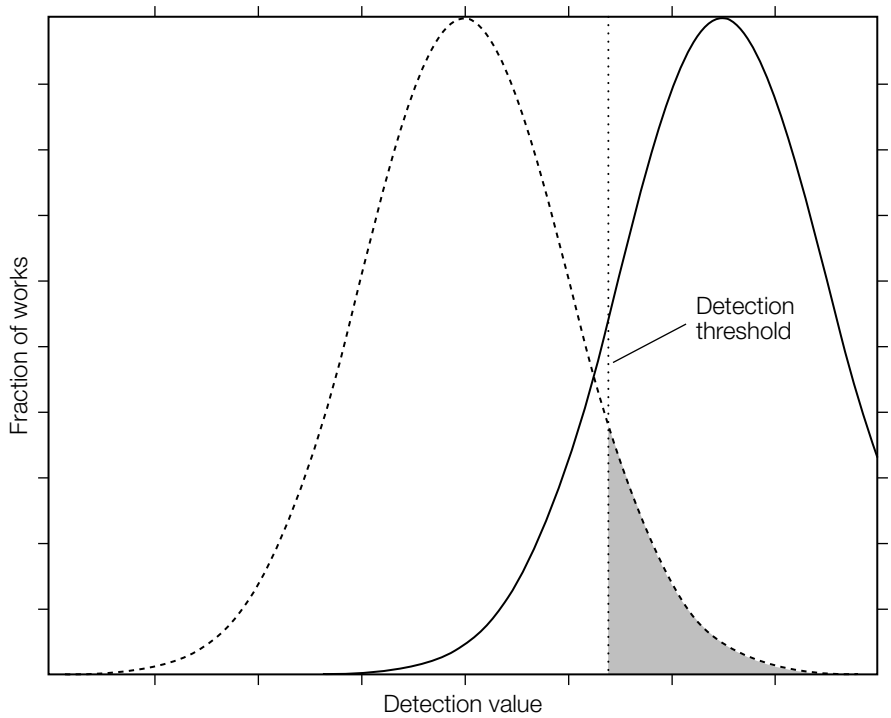


FIGURE 7.2

Example detector output distributions and a detection threshold. The shaded area represents the probability of a false positive.

The shaded area underneath the curve represents the probability of a false positive. The magnitude of this probability depends on the threshold value and the shape of the curve. Thus, modeling the false positive probability can be reduced to modeling the shape of this curve. Several models are presented in this chapter.

The false positive model depends on the watermark detection algorithm and the manner in which the detector is used. In particular, the detector may be looking for one of many watermarks in a single Work, a single watermark in many Works, or many watermarks in many Works. In the first case, we can consider the watermark to be a random variable. In the second case, the Work is the random variable, and in the third case, both the watermark and the Work are considered random. This leads to the concepts of *random-watermark false positive* and *random-Work false positive*.

For example, copy control applications require the detector to search for a small set of *a priori* known watermarks in millions of different Works. In this scenario, the probability of obtaining a false positive depends on the specific reference marks used and on the distribution of unwatermarked content (i.e., it is the Works that are random). On the other hand, in the DiVX application (see Section 2.1.4 of Chapter 2), it was envisioned that there would be millions of DiVX players, and each would embed a unique watermark into the video stream. When a pirated copy of a video was discovered, DiVX intended to determine the source of the illegal copy by searching for the presence of any one of these millions of watermarks in the pirated video Work. In this scenario, the watermarks can be considered random and, because the Works to be examined are also *a priori* unknown, these too are best modeled as a random variable. To the best of our knowledge, the case of a constant Work and random watermarks does not accurately model any real application. However, the random-watermark false positive is reported extensively in the literature and is of theoretical interest. It is important for an application to identify whether the watermark and/or the Work is random, in that watermarks and Works usually have very different distributions. The random-watermark false positive probability and random-Work false positive probability are now discussed in more detail.

7.2.1 Random-Watermark False Positive

The random-watermark false positive probability models the watermark as a random variable and treats the Work as a constant. This is the situation in which the Work or Works are all known *a priori* and in which the number of watermarks is very large and is drawn from a random distribution. Such a scenario most closely models the situation for transaction watermarks. In transaction applications, the number of different watermarks is potentially very large, in that one is needed to identify each customer. When an illicit Work is discovered, the watermark detector must look for the presence of one of these watermarks in the Work in question. However, unless the set of Works to be watermarked was known during the design of the watermarking algorithm,

each Work searched should be considered to be drawn from a distribution of Works.

Although there are few if any real applications for which the false positive error is accurately represented by the random-watermark false positive probability, it is nevertheless common to find this case experimentally examined in the literature. This is because the theoretical analysis is easy and the experimental verification is very straightforward. The detector output distribution is primarily determined by the distribution from which the randomly chosen watermark reference patterns are drawn. Because this distribution is directly under our control, the detector output distribution can be accurately modeled. The experimental verification can be performed without the need for a large number of images. In fact, only one is required. Instead, a large number of watermarks are needed, but these can be dynamically generated using a pseudo-random number (PN) generator.

Experimental results of random-watermark false positive tests were often displayed as shown in Figure 7.3, where the *x*-axis represents many randomly selected reference vectors and the *y*-axis shows the resulting detection value.

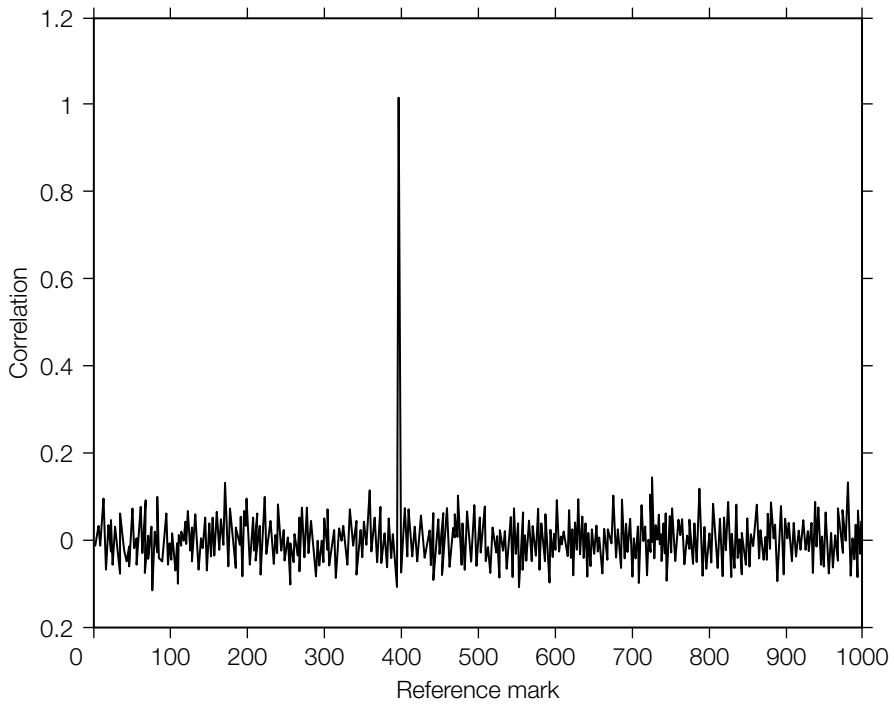


FIGURE 7.3

A commonly used illustration of the ability of a watermark detector to discriminate between reference marks. This Work was watermarked with the reference mark associated with location 400 on the *x*-axis, where the linear correlation value is very high.

Here, the large difference between the value obtained from the one reference vector originally embedded (shown at location 400 on the *x*-axis) and the other reference vectors is presented as a demonstration of the ability of the watermarking system to distinguish between reference marks. However, although this plot provides qualitative information on the ability of the watermark detector to differentiate between watermarks, we prefer a more informative and quantitative plot of the false positive probability as a function of detection threshold, as shown in Figure 7.4.

7.2.2 Random-Work False Positive

Now consider the situation in which many different Works are being examined for the presence of one or a small number of watermarks. In this case, we characterize the *Works* as a random variable and refer to the chance of finding a specific watermark in an unwatermarked Work as the random-Work false positive probability.

This is the probability we are most concerned with in many common applications of watermarking. For example, in a copy control application all possible music or video must be examined for the presence of a small number of watermarks that indicate the recording and playback permissions assigned to a Work. In this case, it is natural to consider the watermarks as constant and the enormous variety of possible Works as a random variable.

Modeling the random-Work false positive probablity requires an accurate model of the distribution of unwatermarked content. This can be very difficult to model. For example, in the E_BLIND/D_LC system (System 1), in which we simply add a globally attenuated reference pattern to the Work in media space, the detector output distribution will depend almost directly on the distribution of Works. In contrast, in the E_BLK_BLIND/D_BLK_CC (System 3), where the image is reduced to a single composite 8 × 8 block and detection is performed using the correlation coefficient, the detector output distribution is determined by the distribution of composite 8 × 8 blocks. This distribution may be very different from the distribution of underlying Works.

The difference between random-watermark and random-Work false positives is highlighted in the following Investigation. In particular, we see that a Gaussian model of the detector output distribution can be suitable for a random-watermark application, but does not satisfactorily model the detector output distribution for a random-Work application.

INVESTIGATION

Examining the Difference between Random-Work and Random-Watermark False Positive Behaviors

In this Investigation, we examine the false positive behavior for linear correlation detection under both random-watermark and random-Work conditions. We first

develop a simple model that assumes that the random distribution is a Gaussian. Two experiments are then performed. The first applies the detector to three Works and searches for the presence of 20,000 random watermarks. The second experiment looks for just three watermarks in 20,000 images (i.e., the Works are random).

Our experiments were performed using the D_LC linear correlation detection algorithm of System 1. Thus, the detector output, z_{lc}, is given by

$$z_{lc} = \frac{1}{N} \sum_i \mathbf{w_r}[i]\mathbf{c}[i], \tag{7.10}$$

where $\mathbf{w_r}$ is a watermark reference pattern and $\mathbf{c}$ is a Work. Equation 7.10 has denoted the watermark as the random vector, and the Work as a constant. For convenience, we will make this random-watermark assumption in the derivation of the false positive probability. Of course, for the random-Work case, the situation is reversed.

Each $\mathbf{w_r}[i]\mathbf{c}[i]$ is a random value drawn from a similar distribution to $\mathbf{w_r}[i]$, but scaled by a value, $\mathbf{c}[i]$. Assuming that the central limit theorem [184] holds, z_{lc} will have a Gaussian distribution with mean, μ_{lc},

$$\mu_{lc} = \mu_{\mathbf{w}}\mu_{\mathbf{c}} \tag{7.11}$$

and standard deviation, σ_{lc}, given by

$$\sigma_{lc} = \sigma_{\mathbf{w}}|\mathbf{c}|. \tag{7.12}$$

If the watermark vectors are chosen to have a mean of zero, the detector output, z, will also have a mean value of zero. The probability that the detector will output a value of x is then given by

$$P_z(x) = \frac{1}{\sqrt{2\pi}\sigma_{lc}} \exp\left(\frac{-x^2}{2\sigma_{lc}^2}\right) \tag{7.13}$$

and the probability of a false positive is

$$P_{fp} = \int_\tau^\infty P(x)\,dx = \int_\tau^\infty \frac{1}{\sqrt{2\pi}\sigma_{lc}} \exp\left(\frac{-x^2}{2\sigma_{lc}^2}\right) dx \tag{7.14}$$

$$= \text{erfc}\left(\frac{\tau}{\sigma_{lc}}\right), \tag{7.15}$$

where erfc denotes the complementary error function defined in Equation 7.8.

The linear correlation algorithm uses the D_LC detector, which is actually testing for the presence of two watermarks: either $\mathbf{w_r}$, which means the watermark

message is 1, or $-\mathbf{w_r}$, which means the message is 0. Thus, with each run of the detector we have two opportunities to obtain a false positive.
The random-watermark false positive probability, $P_{fp(D\_LC)}$, is therefore given by

$$P_{fp(D\_LC)} = \text{erfc}\left(\frac{\tau_{lc}}{\sqrt{2}\sigma_{lc}}\right). \tag{7.16}$$

Experiment 1

The D_LC linear correlator was applied to three images, using 20,000 different reference marks. Each reference mark was generated by drawing its elements independently from a Gaussian distribution, and normalizing the result to have zero mean and unit length. Figure 7.4 shows the resulting false positive rates for each image, as a function of the detection threshold τ_{lc}. The measured rates are indicated by the stars in the graphs. The rates predicted with Equation 7.16 are shown with solid lines.

As we can see from these curves, whereas the false positive behavior varies slightly from one image to the next, the false positive prediction is quite accurate in this range of detection threshold. The variability from one image to the next is due to differences in the standard deviation of the images. This dependence is modeled in Equation 7.12. Given that the *measured* false positive rates fall to zero at high thresholds, we cannot directly assess the accuracy of the model at very low false positive probabilities.

Experiment 2

If we now consider the random-Work false positive probability, we might expect very similar results. After all, we are still applying Equation 7.10; only now, $\mathbf{c}$ is the random variable. However, the elements of $\mathbf{c}$ are the individual pixels of the image, and these are known to be highly correlated spatially. We illustrate the impact of this dependence by considering the random-Work false positive behavior for three different reference marks, each having the same length but different directions. The results presented in the following are generated using the same D_LC linear correlation detector.

The three different reference marks we constructed are shown in the left-hand column of Figure 7.5. These reference marks differ only in their direction in media space. If the simple Gaussian model is accurate, we would expect identical false positive curves.

To test this, we applied the D_LC linear correlation detector to 20,000 images using each of the three reference marks. For each reference mark, we plot the false positive frequency found using a variety of thresholds. The solid line represents the predictions. The reference marks can be seen to have wildly different false positive behaviors. The range of detection values depends on how similar the patterns are to naturally occurring images. (See Linnartz *et al*. [268] for a detailed study of how

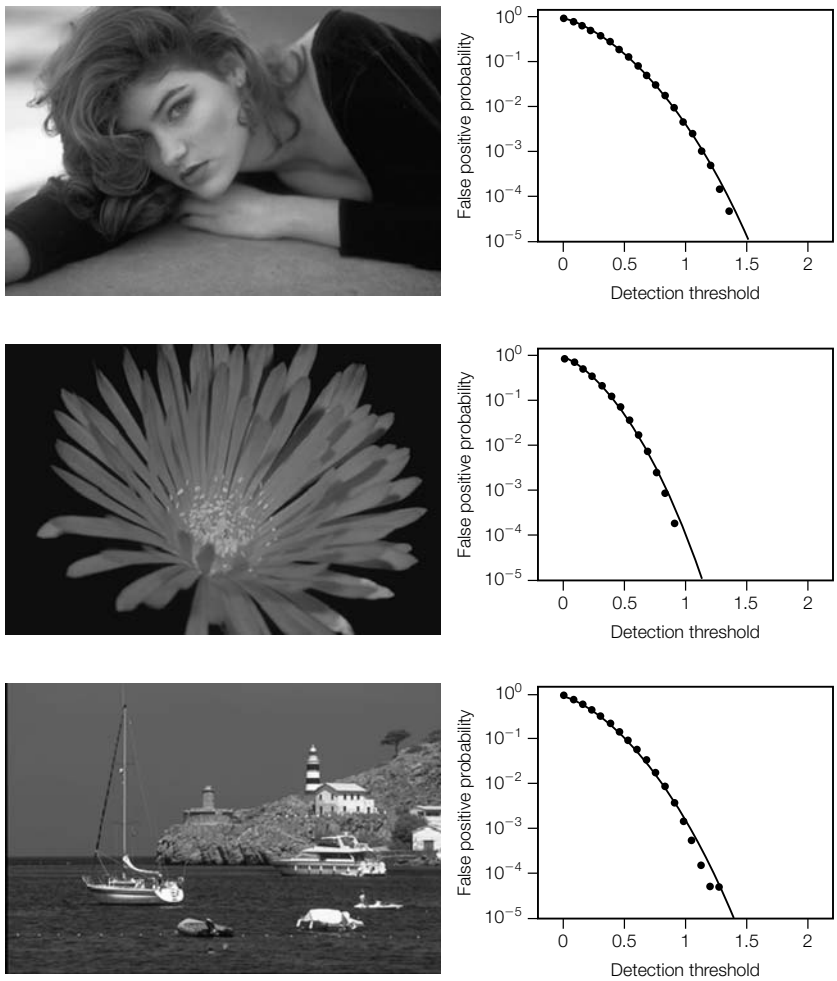


FIGURE 7.4

Random-watermark false positive probabilities for three different images at varying detection thresholds applied to linear correlation. Each image was tested against 20,000 reference marks. The solid lines are the theoretical predictions. The points are the measured results.

the design of watermark reference patterns affects error probabilities in a simple image watermarking system.)

The Gaussian model does not accurately predict the random-Work false positive probability even though it is a very good model of the random-watermark false positive probability. Clearly random-Work and random-watermark false positive probabilities are not the same.

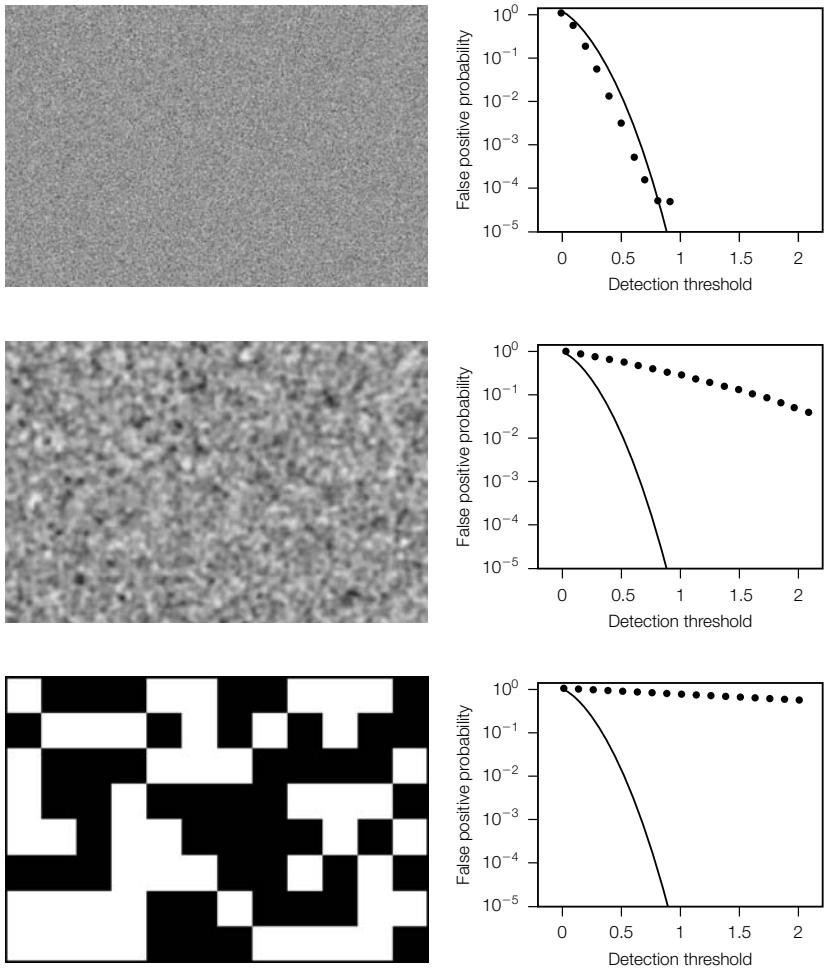


FIGURE 7.5

Random-watermark false positive probabilities for three different reference patterns at varying detection thresholds applied to linear correlation. The graphs were calculated using 20,000 images from the Corel database.

7.3 FALSE NEGATIVE ERRORS

A false negative occurs when a watermark detector fails to detect a watermark that is present. The *false negative probability* is the statistical chance that this will occur, and the *false negative rate* measures the frequency of occurrences.

Figure 7.6 indicates that a false negative occurs because the detector output distribution, represented by the right-hand curve, intersects the threshold τ. Thus, there is a finite possibility that the detector output will be less than the threshold, τ, even when a watermark is present in the Work.

An analysis of the false negative probability can follow the same lines as that for the false positive probability. Again, we can identify the exact types of false negative probabilities that are most relevant to a given application. Once more we can make a distinction between random-watermark and random-Work probabilities. If we are going to embed a million different watermarks into different copies of a small number of Works, we are most interested in random-watermark false negative probabilities. If we are going to embed a small number of different watermarks into a million Works, we should examine random-Work probabilities.

However, unlike the case of false positive probabilities, there are many more variables to consider before analyzing the probability of a false negative. This is because false negative probabilities are highly dependent on both

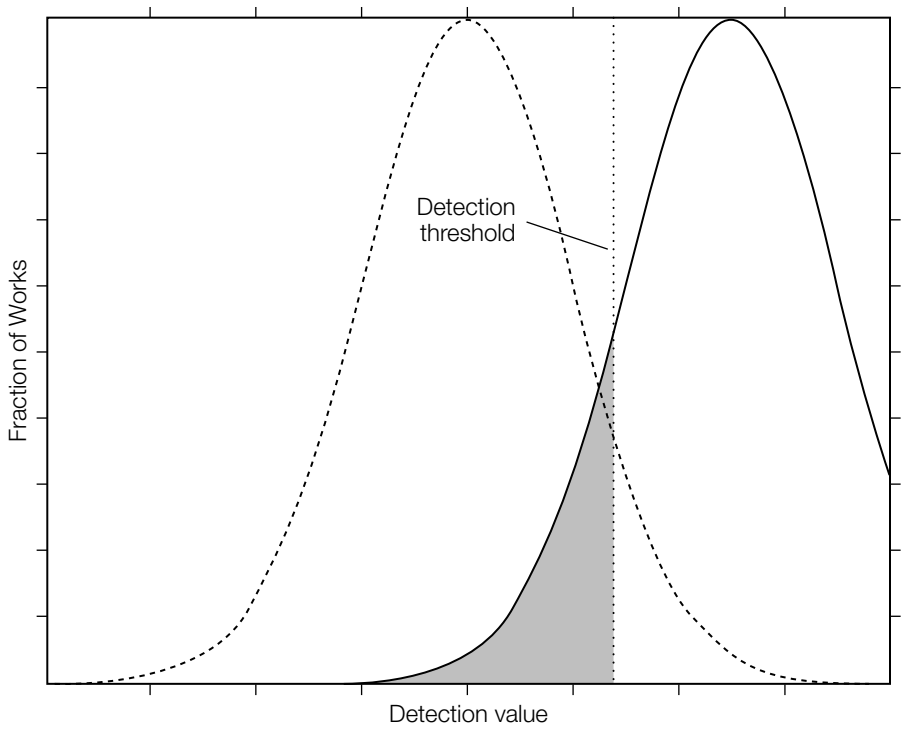


FIGURE 7.6

Example detector output distributions and a detection threshold. The shaded area represents the probability of a false negative.

the watermark detector *and* the embedder, and what happens to a Work between the time a watermark is embedded and the time it is detected. A watermark might be severely distorted by filtering, analog transmission, lossy compression, or any of a wide variety of processes, thus increasing the probability of a false negative. It also might be attacked by some adversary, using an algorithm specifically designed to result in nearly 100% false negatives. Before beginning to estimate the false negative probability, we must specify what processes are expected between embedding and detection.

The possible choices of expected processes can be divided into three main categories, each of which corresponds to a property of watermarking systems discussed in Chapter 2:

- The *security* of a system is its ability to survive an attack by hostile adversaries. In many cases, the greatest security concern is that an adversary might make the watermark undetectable. Modifications of a Work for the purpose of defeating a watermark detector are known as unauthorized removal attacks.

  A watermarking system's level of security against these attacks can be thought of as the probability of detection after an attack by a random adversary. This is just the complement of the probability of a missed detection (false negative) after attack by a random adversary. That is, the level of security is $1 - P_{\text{fn}}$. (Of course, it is extremely difficult to quantify this probability.)

- The *robustness* of a system is its ability to survive normal processing, such as filtering, analog transmission, and lossy compression. Robustness can vary wildly from one type of processing to another. For example, a watermark that is highly robust to analog transmission might be completely eliminated by lossy compression. Thus, it is usually best to talk about a system's robustness *to a specific process*.

  A watermarking system's level of robustness to some process can be thought of as the probability the watermark will be detected after that process is applied. This is just the opposite of the probability of a false negative after application of the process (i.e., $1 - P_{\text{fn}}$).

- The *effectiveness* of a system is its ability to embed and detect watermarks when there is *no* intervening attack or processing. This is just the complement of the probability of a false negative immediately after the watermark is embedded (i.e., $1 - P_{\text{fn}}$).

Robustness and security are discussed in Chapters 9 and 10, respectively. The effectiveness of watermarking systems is addressed analytically in Section 7.6.2 for a system similar to the E_BLK_BLIND/D_BLK_CC watermarking system (System 3). Empirical results for many of the example watermarking systems are presented throughout the book.

7.4 ROC CURVES

In every watermarking system there is a tradeoff between the probability of false positives and the probability of false negatives. As the threshold increases, the false positive probability decreases and the false negative probability rises. The performance of the system can only be interpreted by considering both probabilities at once. In this section, we describe the *receiver operating characteristic curve,* or *ROC curve,* which is a standard graphical tool for interpreting the performance of communications systems [184].

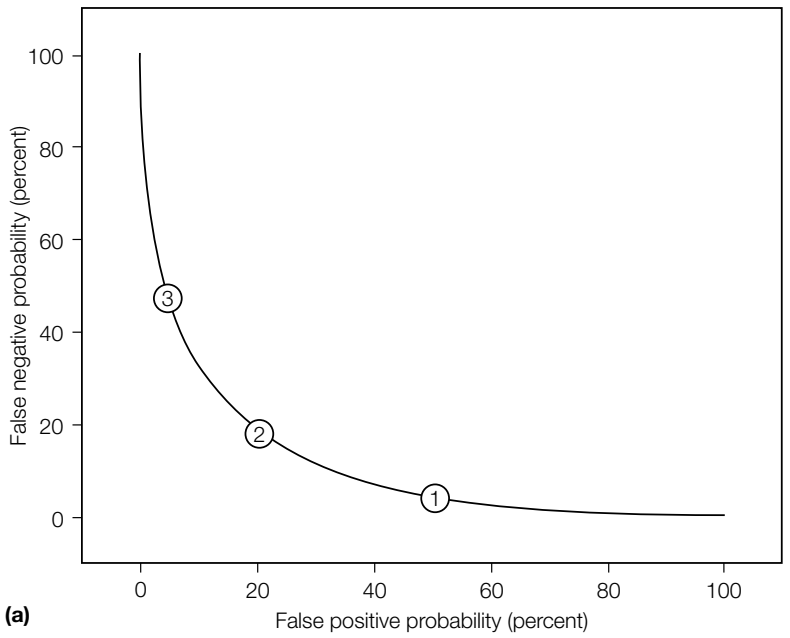
To this point, we have shown our experimental results by plotting histograms of detection values. These provide an intuitive picture of the results. If the histogram of the detection values obtained from unwatermarked content (which we usually plot with dashed lines) is well separated from the detection values obtained from watermarked content (usually solid lines), then it is clear that many thresholds between them will yield both low false positive probabilities and low false negative probabilities. Figures 3.6, 3.11, and 3.20 in Chapter 3 all indicate such thresholds at the tops of the graphs.

ROC curves are an alternative to these histogram plots. An ROC curve is a parametric curve that plots the false positive probability (usually the x-axis) against the false negative probability (usually the y-axis) as a function of threshold.

The generation of an ROC curve for a real watermarking system usually requires the use of a theoretical model of either the false positive or false negative behavior of the system. To illustrate this point, we first present an ROC curve of a hypothetical watermarking system that has very poor performance. The purpose of this illustration is to examine the relationship between the ROC curve and the type of histograms used in Chapter 3. We then examine a histogram from one of the real watermarking systems presented in Chapter 3. We find that there is not enough data to generate a useful ROC curve. This is the typical case for a real watermarking system. To remedy this, we propose building a model of the false positive behavior, the false negative behavior, or both, and using these models to interpolate the required data. In the example presented, we model the false positive behavior and use the measured false negatives to generate an ROC curve.

7.4.1 Hypothetical ROC

Figure 7.7(a) shows an example, based on a hypothetical watermarking system. For comparison, a normalized histogram of the same information is shown in Figure 7.7(b). This plots the relative frequencies of detection values for unwatermarked and watermarked content. To help compare the two plots, we have labeled three possible threshold values on each of them. The three marks on the curves in Figure 7.7(b) correspond to the three marks on the curve in Figure 7.7(a). With the threshold set to correspond to the point marked 2 in



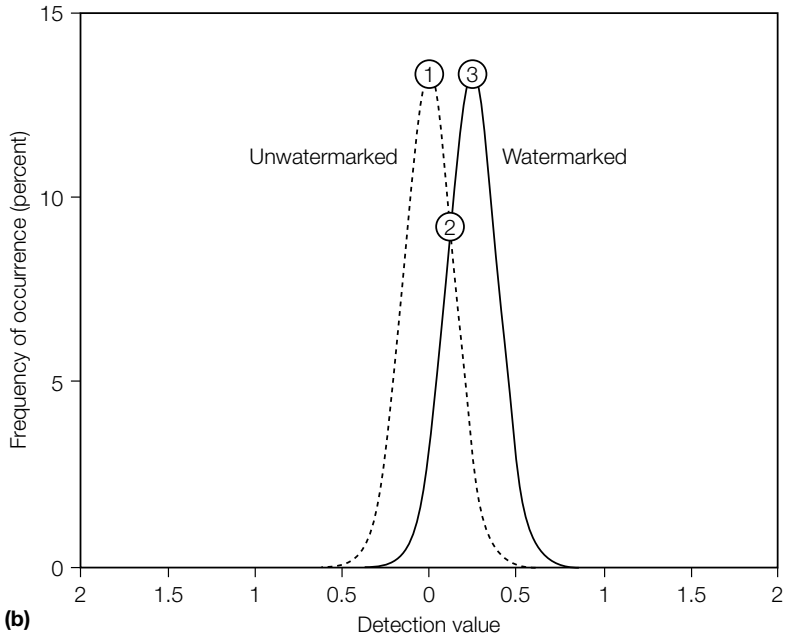


FIGURE 7.7

ROC curve (a) and normalized histogram (b) for a hypothetical watermarking system. The numbered circles indicate corresponding threshold values.

the normalized histogram, the false positive probability is equal to the false negative probability. This fact is reflected in the location of point 2 in the ROC curve. With the threshold set to the detection value corresponding to the point marked 1, the false positive probability is 0.5 and the false negative probability is low. Similarly, at 3, the false negative probability is 0.5 and the false positive probability is low.

The hypothetical watermarking system shown in Figure 7.7 has very poor performance. A quick glance at either of the two graphs shows that to obtain a reasonable false positive probability we must choose a threshold that leads to a high probability of false negatives. In real watermarking systems it should be possible to find thresholds that lead to very small probabilities of either type of error.

7.4.2 Histogram of a Real System

Consider the histogram plotted in Figure 7.8. This is based on real experimental data; namely, the results obtained in Chapter 3 for the E_BLIND/D_LC

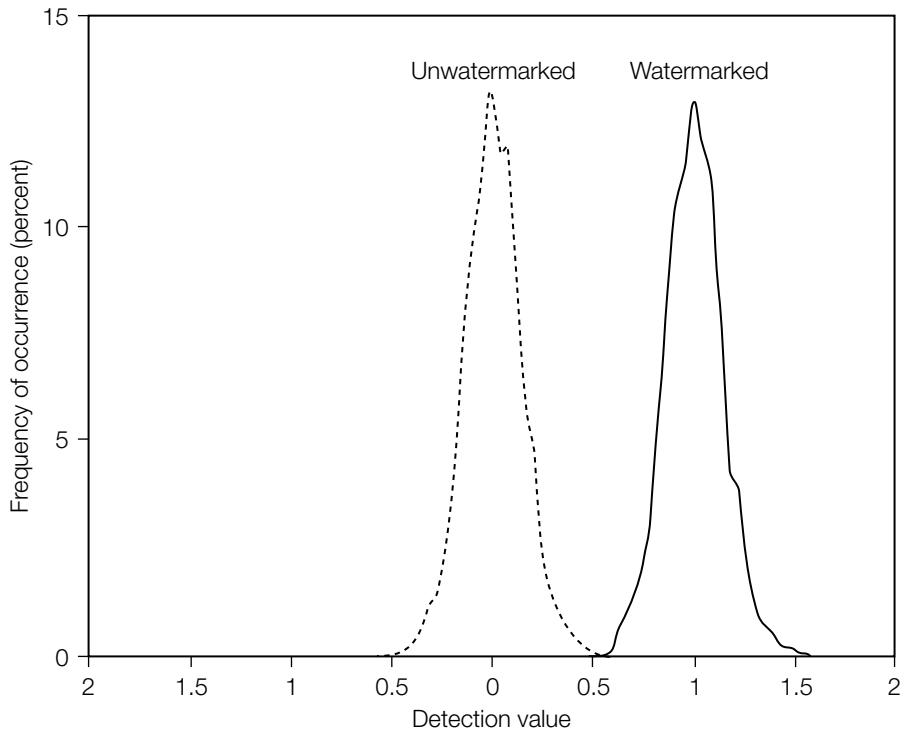


FIGURE 7.8

Normalized histogram for the E_BLIND/D_LC watermarking system.

watermarking system (System 1) on unwatermarked images (dashed line) and on images marked with a message of $m = 1$ (solid line).

The problem with the empirical data plotted here is that during the test there was only a very small overlap between detection values obtained from watermarked and unwatermarked images, which was not enough to make a meaningful ROC curve. The lowest detection value obtained from a watermarked image was 0.46. The highest value from an unwatermarked image was 0.60. Thus, if we were to plot an ROC curve with this data, the false negative rate would be exactly 0 for any threshold below 0.46, and the false positive rate would be 0 for any threshold above 0.60. The curve would be vertical on the left, horizontal on the right, and only informative in a very small region near the lower left corner. ROC curves are informative in providing a graphical representation of the tradeoff between false positive and false negative behavior, but in this case there is almost no tradeoff and we cannot generate a useful ROC curve.

7.4.3 Interpolation Along One or Both Axes

Of course, even though we did not obtain a detection value over 0.60 from unwatermarked data, we cannot conclude that we will never obtain one. In theory, if we were to test enough unwatermarked images, we would probably find some with quite high detection values, allowing us to measure the frequencies with which those high values arise and plot a more meaningful ROC curve. However, such an experiment would require an impractical amount of computation.

Instead, a meaningful ROC curve can be plotted by estimating one or both of the probabilities along the axes. For example, we might model the probability of a false positive at high thresholds. This model would then give us nonzero false positive probabilities at thresholds that yield nonzero false negative rates in our test. The validity of an ROC curve plotted in this way depends on the accuracy of our false positive probability model.

In cases where it is difficult to estimate error probabilities analytically, we can make our estimates by extrapolating empirical data. The experiment that generated the data plotted in Figure 7.8 tested random-Work probabilities (the detector was applied to 2,000 unwatermarked images and 2,000 watermarked images, all using the same reference mark). As previously discussed, it is difficult to estimate the random-Work false positive probabilities, because the images are drawn from a distribution that is not white and Gaussian. However, if we assume that the distribution of detection values is approximately Gaussian, we can estimate the false positive probabilities by computing the sample mean and variance of the detection values obtained from unwatermarked data. These are then used in the complimentary error function. Figure 7.9 shows an ROC curve computed in this manner from the data used in Figure 7.8. Because the false positive probabilities at higher thresholds are extremely low, we have made this plot with a logarithmic x-axis.

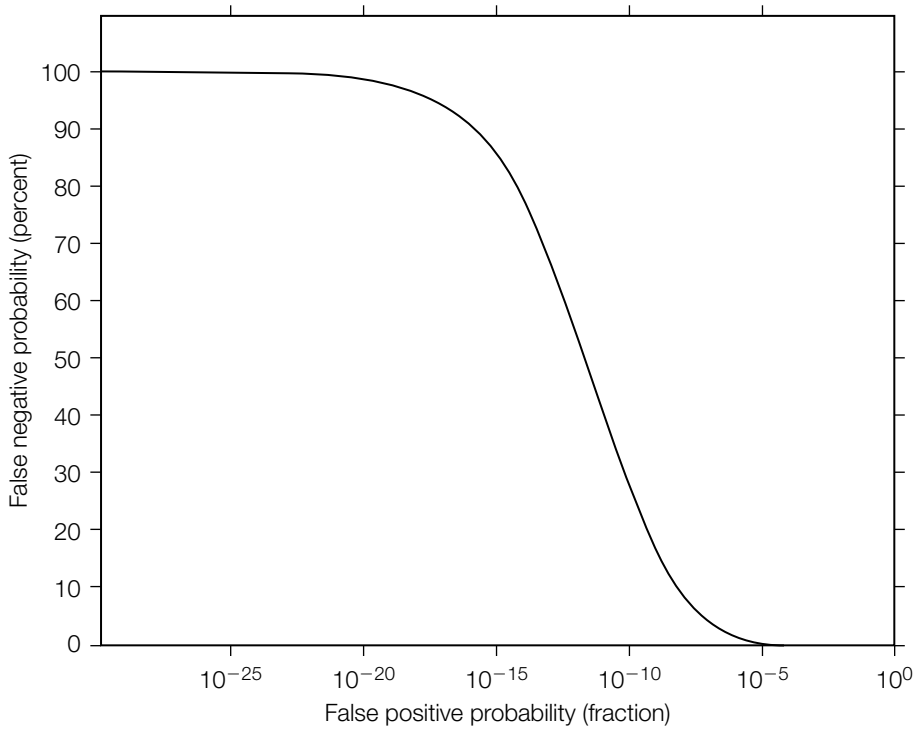


FIGURE 7.9

ROC curve for the E_BLIND/D_LC watermarking system. The false positive probability for the ROC curve has been estimated by fitting a Gaussian distribution to the test results for unwatermarked images (dashed line in Figure 7.8), and the *x*-axis is plotted on a logarithmic scale.

7.5 THE EFFECT OF WHITENING ON ERROR RATES

The model used in the Investigation of Section 7.2 assumes that the random vectors are additive white Gaussian. When a signal is corrupted by AWGN noise, linear correlation is an optimum method of detection. However, if linear correlation is used when the noise is *not* additive white Gaussian, detection will be suboptimal and the detector output will not behave as predicted. This situation occurred when we applied the D_LC detector to random Works. We observed that for the random-Work false positive estimation our simple model breaks down. This is often the case for random-Work false positives, in that most forms of content (be it music, photographs, or movies) cannot be modeled by an independent white Gaussian distribution. The true distribution of unwatermarked content is more complex.

Multimedia content typically exhibits strong spatial and/or temporal correlations. For example, image pixels in close proximity to one another typically have similar brightness. Temporal correlations exist in audio and video signals. A common approach to processing correlated data is to first decorrelate the data and then apply signal-processing algorithms that are optimum for white noise. Decorrelation can be accomplished through the application of a *whitening* or *decorrelating* filter, which is designed to eliminate or at least reduce the correlation present. Thus, a whitening filter can significantly improve the performance of detectors that assume the noise is uncorrelated.

To use a whitening filter, $\mathbf{f_{wh}}$, in a linear correlation watermark detector we first convolve both the received Work, $\mathbf{c}$, and the reference pattern, $\mathbf{w_r}$, with $\mathbf{f_{wh}}$, and then compute the correlation between the resulting patterns. That is,

$$z_{\text{wh}}(\mathbf{c}, \mathbf{w_r}) = z_{\text{lc}}(\mathbf{c}', \mathbf{w}_\mathbf{r}'), \tag{7.17}$$

where

$$\mathbf{c}' = \mathbf{c} * \mathbf{f_{wh}} \tag{7.18}$$

$$\mathbf{w}_\mathbf{r}' = \mathbf{w_r} * \mathbf{f_{wh}}, \tag{7.19}$$

and $*$ indicates convolution.

Whitening filters are a standard tool in designing signal detectors. The use of such a filter for image watermarking was first suggested by Depovere *et al.* [111]. They pointed out that a simple, horizontal difference filter applied to the rows of an image would remove most of the correlation between horizontally adjacent pixels. This difference filter yields acceptable results when applied directly to images. However, it is not necessarily applicable to other media. It is also not necessarily applicable to vectors that have been extracted from Works using various other processes.

A more general approach to designing whitening filters can be applied if we assume that the elements of unwatermarked Works are drawn from a correlated Gaussian distribution. That is, we assume the probability of obtaining $\mathbf{c_o}$ is given by

$$P(\mathbf{c_o}) = \frac{1}{(\sqrt{2\pi})^N \sqrt{\det(\mathbf{R})}} \exp\left(-\frac{(\mathbf{c_o} - \mu_{\mathbf{c_o}})^{\mathrm{T}} \mathbf{R}^{-1} (\mathbf{c_o} - \mu_{\mathbf{c_o}})}{2} \right), \tag{7.20}$$

where $\mathbf{R}$ is a covariance matrix. In this case, the whitening filter can be obtained by extracting the center row from the matrix square root of inverse of $\mathbf{R}$, (i.e., $\sqrt{\mathbf{R}^{-1}}$).

The derived filter whitens vectors drawn from the distribution described by **R**. However, if this filter is applied to vectors drawn from another distribution, it will not have a whitening effect. In fact, if it is applied to vectors drawn from a distribution that is already white, it will have the effect of *introducing* correlations. A consequence of this is that whitening often has opposite effects on random-Work and random-watermark false positive probabilities. Suppose the watermarks are drawn from a white Gaussian distribution, whereas the Works are drawn from some correlated distribution. Without whitening, the random-watermark false positive probability is easily predicted, but the random-Work false positive probability is more complex. Applying a whitening filter designed to decorrelate the Works makes the random-Work false positive probability behave better, but causes the random-watermark false positive probability to become badly behaved. When we are more concerned with random-Work false positive probabilities than with random-watermark false positive probabilities, as is the case in most applications, the use of whitening filters is usually justified.

It must also be noted that the tails of the distribution of detection values are greatly affected by any mismatch between the Gaussian model and the true distribution of unwatermarked media. For example, suppose we are using a whitening filter in a video watermark detection system. We have derived the filter based on a simple correlated Gaussian model of the distribution of unwatermarked frames. This model might be reasonably accurate for normal shots of actors, scenery, and so forth. However, every now and then a film contains a short closeup of a television screen showing only video noise. Video noise is largely uncorrelated, and therefore the whitening filter will introduce correlations into these scenes. The result may be a higher chance of a false positive.

Nevertheless, whitening can be a useful tool when we are either unconcerned with the tails of the distribution of detection values, have a limited set of unwatermarked Works entering the system (e.g., no shots of video noise), or employ an extraction process that results in a nearly Gaussian distribution.

INVESTIGATION

Whitened Linear Correlation Detection

Our purpose here is to illustrate both the derivation and use of a whitening filter. We therefore choose not to use the approximate whitening filter suggested in Depovere *et al*. [111] (namely, $\mathbf{f}_{wh} = [-1, 1]$). Instead, we derive a more sophisticated whitening filter based on the assumption, found in Linnartz *et al*. [268], that the correlation between two pixels is proportional to the Manhattan distance between them. When this assumption is accurate, our whitening filter will be optimum.

System 11: E_BLIND/D_WHITE

To derive the whitening filter, we assume an elliptical Gaussian model of the distribution of unwatermarked images, using the covariance matrix suggested by

Linnartz *et al.* [268]. This covariance matrix is given as

$$\mathbf{R}[i_1, i_2] = q^{|x_1 - x_2| + |y_1 - y_2|} \tag{7.21}$$

for every pair of pixels, i_1, i_2, where (x_k, i_k) is the position of pixel i_k, and q is a constant. Linnartz *et al.* suggest that q should lie between 0.90 and 0.99. We use $q = 0.95$. Substituting the resulting covariance matrix into Equation 7.20 yields an elliptical Gaussian distribution.

Because $\mathbf{R}[i_1\, i_2]$ drops off fairly rapidly with the distance between i_1 and i_2, we can find all significant values of $\mathbf{R}$ within a limited pixel window. We therefore concern ourselves only with the correlations between pixels in an 11×11 window. Thus, the matrix $\mathbf{R}$ has dimensions 121×121.

To find the whitening filter, we first compute the matrix square root of the inverse of $\mathbf{R}$. This results in the matrix $\mathbf{g}_{\text{wh}}$. The rows of this matrix are approximately the same as one another, apart from a shift. The central row, row 61, gives us a reasonable whitening filter, $\mathbf{f}_{\text{wh}}$. Reshaping this row to an 11×11 matrix yields the filter shown in Table 7.1 (rounded to the nearest tenth).

The system investigated here uses the E_BLIND blind embedder first introduced in System 1. The D_WHITE detection algorithm is essentially the same as the D_LC algorithm, except that it convolves the image, $\mathbf{c}$, and the reference mark, $\mathbf{w_r}$, by $\mathbf{f}_{\text{wh}}$ before computing the linear correlation. The reference mark is normalized to have unit variance after filtering. Thus, the detector computes

$$z_{\text{wh}}(\mathbf{c}, \mathbf{w_r}) = \frac{1}{N}\left(\mathbf{f}_{\text{wh}} * \mathbf{c}\right) \cdot \left(\frac{\mathbf{f}_{\text{wh}} * \mathbf{w_r}}{s_{\text{wh}}}\right), \tag{7.22}$$

Table 7.1 Whitening filter used in D_WHITE detection algorithm.

0.0	0.0	0.0	0.0	0.1	−0.2	0.1	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.1	−0.3	0.1	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.2	−0.5	0.2	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.1	0.4	−1.1	0.4	0.1	0.0	0.0	0.0
0.1	0.1	0.2	0.4	1.8	−5.3	1.8	0.4	0.2	0.1	0.1
−0.2	−0.3	−0.5	−1.1	−5.3	15.8	−5.3	−1.1	−0.5	−0.3	−0.2
0.1	0.1	0.2	0.4	1.8	−5.3	1.8	0.4	0.2	0.1	0.1
0.0	0.0	0.0	0.1	0.4	−1.1	0.4	0.1	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.2	−0.5	0.2	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.1	−0.3	0.1	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.1	−0.2	0.1	0.0	0.0	0.0	0.0

where $\mathbf{c}$ is the input image, $\mathbf{w_r}$ is a predefined reference pattern of the same dimensions as $\mathbf{c}$, and s_{wh} is the sample standard deviation of $\mathbf{w_r} * \mathbf{f}_{wh}$. D_WHITE then uses this value to determine its output, thus:

$$m_N = \begin{cases} 1 & \text{if } z_{wh}(\mathbf{c}, \mathbf{w_r}) > \tau_{wh} \\ \textit{no watermark} & \text{if } -\tau_{wh} \le z_{wh}(\mathbf{c}, \mathbf{w_r}) \le \tau_{wh} \\ 0 & \text{if } z_{wh}(\mathbf{c}, \mathbf{w_r}) < -\tau_{wh}. \end{cases} \tag{7.23}$$

Experiment 1

To evaluate the whitening filter, we compared the performance of the E_BLIND/D_WHITE system with the E_BLIND/D_LC system of Chapter 3. Once again we ran the detector on 2,000 unwatermarked images and the same images after embedding a message, $m = 0$ or $m = 1$.

Figure 7.10 shows the performance obtained using the D_WHITE algorithm to detect watermarks embedded with E_BLIND. This should be compared against the results obtained using a linear correlation detector without whitening (e.g., the DLC detector) on the same watermarked images (Figure 3.6 in Chapter 3). Clearly, the degree of separation between the three cases (no watermark, watermark message = 1, and watermark message = 0) is significantly greater than with the D_LC detector.

Experiment 2

To examine the effect of whitening on false positives, we repeated Experiment 2 of the Investigation in Section 7.2.2 using the D_WHITE detector to look for three different watermarks in 20,000 images.

Figure 7.11 shows the measured random-Work false positive probabilities for three different reference marks. These results should be compared against those shown in Figure 7.5, which were obtained without whitening. Note that with whitening the false positive behaviors of the different patterns are far more similar to one another than without. Thus, whitening allows us greater freedom in the selection of our reference marks.

Figure 7.11 also reveals that the measured false positive probabilities still deviate significantly from the predicted curve. The derivation of the whitening filter assumes that the noise is drawn from a correlated Gaussian distribution. The experiments suggest that this assumption is not accurate. In fact, several studies have suggested that a Laplacian or generalized Gaussian distribution is a more accurate model for images [305, 345].

Experiment 3

Figure 7.12 shows the random-Work false positive probabilities when the Works are not drawn from the natural distribution. Here, the Works are pure random

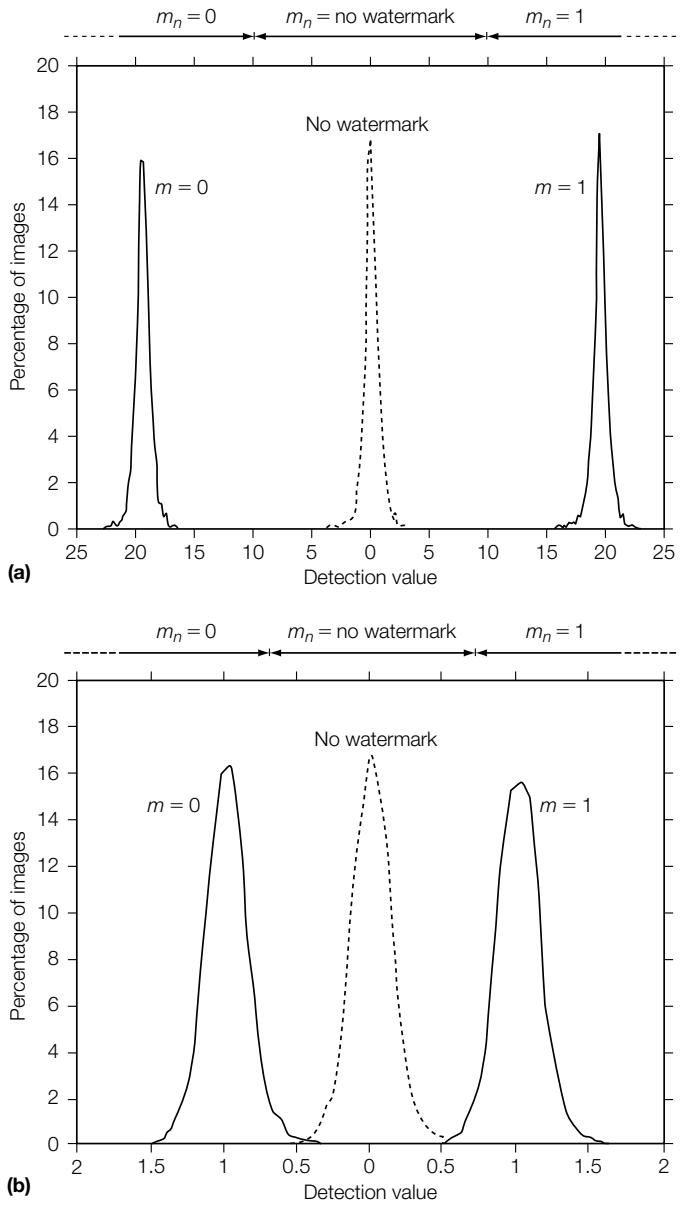


FIGURE 7.10

Comparison of linear correlation detection, with and without whitening, on watermarks embedded with the E_BLIND embedder. (a) Performance of D_WHITE detection algorithm and (b) performance of D_LC detection algorithm, as reported in Figure 3.6.

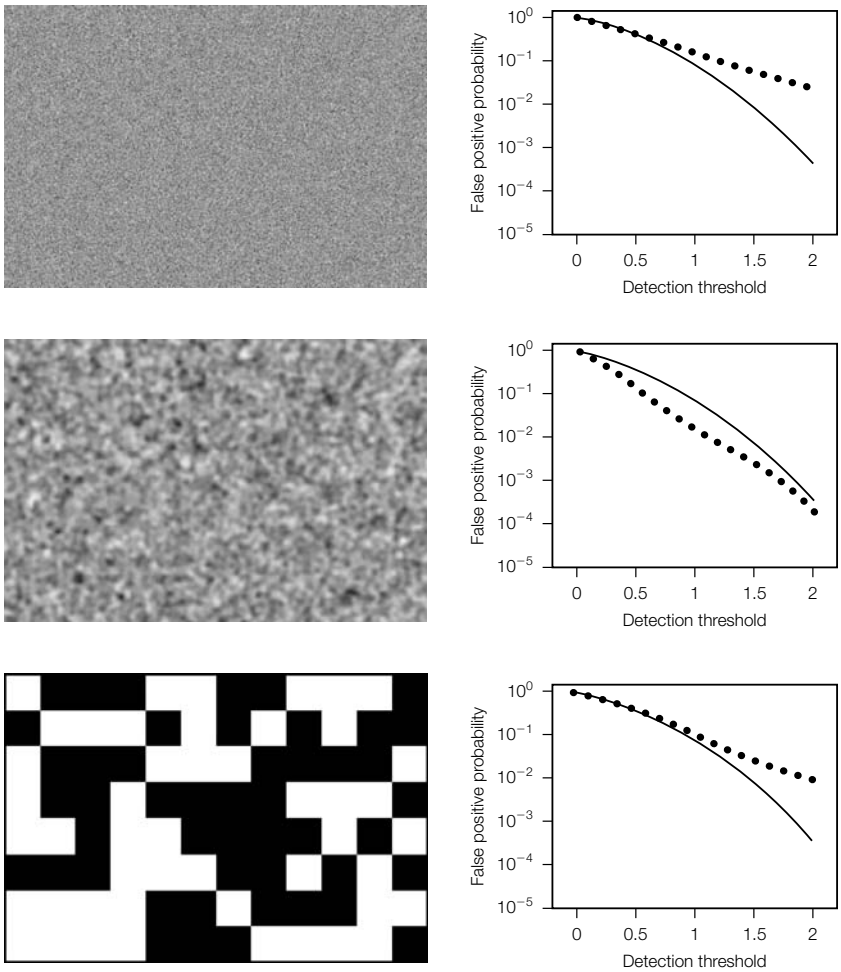


FIGURE 7.11

Random-work false positive probabilities for the D_WHITE detection algorithm, using three different reference marks and various thresholds. The graphs were calculated using 2,000 randomly chosen images from the Corel database.

images, with each pixel drawn independently from a uniform distribution between black and white. This is a rough simulation of the video noise mentioned earlier. The whitening filter introduces correlation into the distribution of vectors that are compared against the filtered reference mark, and the resulting false positive rates are much higher than those illustrated in Figure 7.11. Note that the graphs in these two figures show the same range of thresholds along their *x*-axes. This illustrates one of the risks of using a whitening filter.

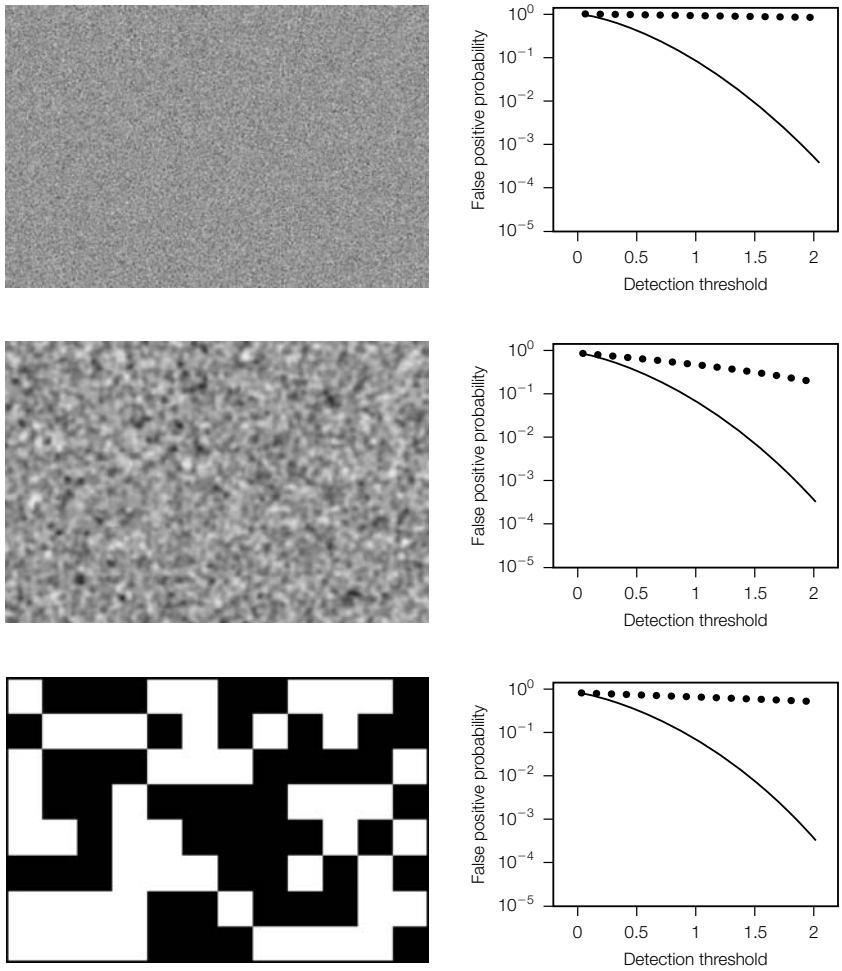


FIGURE 7.12

Random-Work false positive probabilities for the D_WHITE detection algorithm, with 2,000 randomly generated (synthetic) images rather than natural images.

7.6 ANALYSIS OF NORMALIZED CORRELATION

Our discussion of error models has so far focused on the use of linear correlation as a detection statistic. However, there are several other forms of related measures, as described in Section 3.5 of Chapter 3. In particular, normalized correlation, or correlation coefficient, is often preferred because of its robustness to amplitude changes in the Work. Despite the superficial similarity between linear and normalized correlation, the shape of their detection regions is very

different and results in different error rates. Because normalized correlation is one of the most common detection measures, we devote the rest of this chapter to an analysis of the false positive and false negative error to which it leads.

7.6.1 False Positive Analysis

In the Investigation of Section 7.2, we modeled the output from a linear correlator by a Gaussian distribution. Here we examine two models that can be used for normalized correlation.

The approximate Gaussian method, presented first, assumes that the output from the normalized correlation detector has a Gaussian distribution. Because normalized correlation is bounded between ±1, we expect that this model will perform poorly at the tails of the distribution. This is indeed the case. Nevertheless, if our false positive requirement is moderate (i.e., the detection threshold is far enough from these bounds), then the Gaussian approximation may be acceptable.

The second model gives an exact value for the probability of a false positive, under the assumption that the random vectors (watermarks or Works) are drawn from a radially symmetric distribution. The analysis is based on examining the intersection of conical detection regions with the surface of a sphere, both of which are in a very high-dimensional space. The Investigation of the spherical method shows that random-watermark false positives are accurately predicted. For random-Work false positives, a reasonable match between theory and experiment is obtained if whitening is applied during detection.

Approximate Gaussian Method

The simplest method of estimating false positive probabilities is to assume that the distribution of detection values obtained from unwatermarked Works is Gaussian. We refer to this as the *approximate Gaussian method*.

To find the variance of the assumed Gaussian distribution, begin by considering the random-watermark false positive probability for a given Work using a linear correlation detector, with the elements of the reference mark drawn from identical, independent distributions. From Equation 7.12, the standard deviation, σ_{lc}, is

$$\sigma_{lc} = |\mathbf{v}|\sigma_{\mathbf{w_r}}, \tag{7.24}$$

where $\mathbf{v}$ is the extracted vector. Remember that the normalized correlation, z_{nc}, is given by

$$z_{nc} = \frac{\mathbf{v} \cdot \mathbf{w_r}}{|\mathbf{v}||\mathbf{w_r}|}, \tag{7.25}$$

with its standard deviation, σ_{nc}, being

$$\sigma_{nc} = \frac{|\mathbf{v}|\sigma_{\mathbf{w_r}}}{|\mathbf{v}||\mathbf{w_r}|}. \tag{7.26}$$

The magnitude of the reference mark is approximately

$$|\mathbf{w_r}| = \sqrt{N}\sigma_{\mathbf{w_r}}, \tag{7.27}$$

where N is the number of dimensions in mark space. Thus, the variance of the assumed Gaussian is

$$\sigma_{\text{nc}}^2 = \frac{1}{N}. \tag{7.28}$$

This leads to the following estimate of false positive probability, when the normalized correlation is tested against a threshold of τ_{nc}:

$$P_{\text{fp}} \approx \text{erfc}\left(\sqrt{N}\tau_{\text{nc}}\right). \tag{7.29}$$

INVESTIGATION

An Evaluation of the Accuracy of the Approximate Gaussian Method

Here we investigate the accuracy of the random-watermark false positive probability estimated using the approximate Gaussian method. The evaluation is performed using the D_BLK_CC watermark detector, previously described in Chapter 3.

Experiment

To test the accuracy of the approximate Gaussian method for estimating random-watermark false positive probabilities, we ran the D_BLK_CC detector on three different images with one million random watermarks per image. By testing this large number of watermarks, we obtain reasonable observed false positive rates for higher thresholds, where we shall see that the approximate Gaussian approach breaks down.

Figure 7.13 shows the results. The points on the graphs show the observed false positive rates for various thresholds. The solid lines show the estimate obtained from Equation 7.29.[1] The approximate Gaussian method is reasonably accurate when the detection threshold is low. However, as the threshold exceeds 0.4, the approximate Gaussian method begins to overestimate the false positive probability. Clearly, the overestimation will become dramatic when the threshold

[1] Note that these estimates were computed using $N = 63$, even though the D_BLK_CC detector computes its detection value from a 64-dimensional vector (an 8 × 8 summed block). This is because the detector uses the correlation coefficient, which is the same as normalized correlation in a space with one fewer dimension. Note also that the estimate is independent of the Work, so the solid line is the same in each of the graphs.

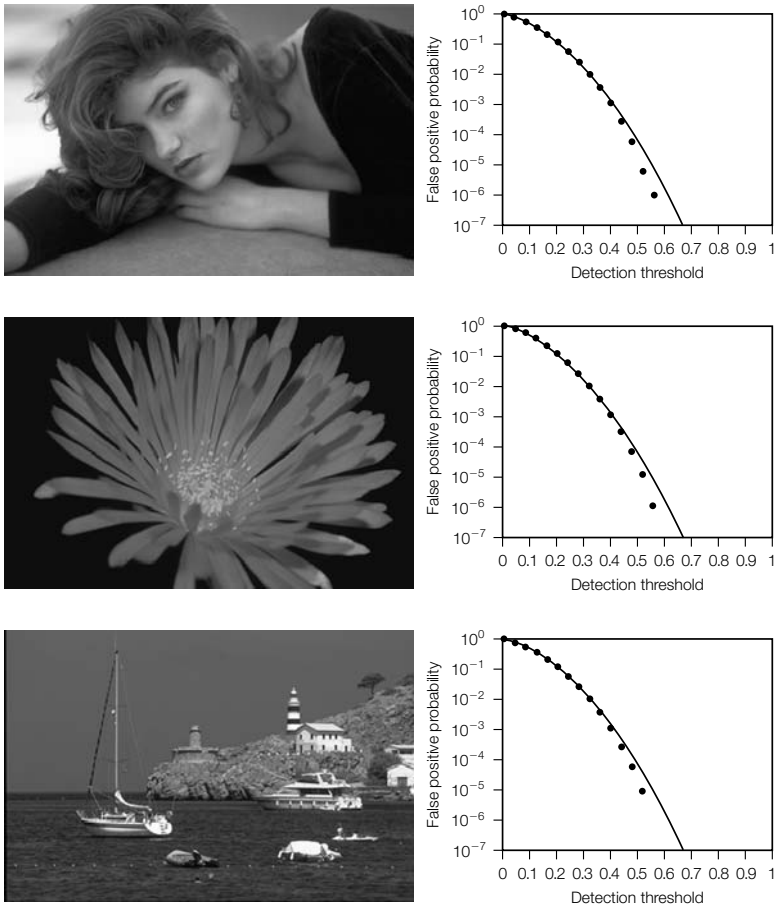


FIGURE 7.13

Random-watermark false positive rates for the correlation coefficient–based D_BLK_CC detection algorithm. Empirical results (points) are for 1,000,000 random reference marks compared against the three illustrated images, using the D_BLK_CC detector. Theoretical estimates (solid lines) are computed using the approximate Gaussian method.

is greater than 1, because in such a case the approximate Gaussian method yields a nonzero estimate, when in truth it is impossible for a normalized correlation value to exceed 1, whether or not a watermark is present. Thus, if our application's requirements are strict enough that the false positive behavior must be accurately modeled at values smaller than 10^{-5}, we need a different method of estimating the false positive probability.

Spherical Method

A more precise method of estimating probability of a false positive when using normalized correlation is described in Miller and Bloom [289] and is presented in Section B.3 of Appendix B. This method, which we refer to as the *spherical method,* gives an exact value for the probability, under the assumption that the random vectors are drawn from a radially symmetric distribution. In Appendix B, we derive the spherical method for the case of a random watermark, $\mathbf{w_r}$, and a constant Work, $\mathbf{c}$. That is, we assume that the probability of obtaining a given random vector, $\mathbf{w_r}$, depends only on the length of $\mathbf{w_r}$ and is independent of the direction of $\mathbf{w_r}$. A white Gaussian distribution satisfies this assumption.

If a random N-dimensional vector is drawn from a radially symmetric distribution, the probability that its normalized correlation with some constant vector will be over a given threshold, τ_{nc}, is given exactly by Miller and Bloom [289]:

$$P_{\text{fp}} = \frac{\int_0^{\cos^{-1}(\tau_{\text{nc}})} \sin^{N-2}(u)\, du}{2\int_0^{\pi/2} \sin^{N-2}(u)\, du}. \tag{7.30}$$

Evaluating Equation 7.30 requires calculation of the integral $I_d(\theta) = \int_0^{\theta} \sin^d(u)\, du$, with $d = N - 2$. This integral has a closed-form solution for all integer values of d. Table 7.2 provides the solutions for $d = 1$ through 5. When $d > 5$,

Table 7.2 Closed-form solutions for $I_d(\theta)$.

d	$I_d(\theta)$
0	θ
1	$1 - \cos(\theta)$
2	$\dfrac{\theta - \sin(\theta)\cos(\theta)}{2}$
3	$\dfrac{\cos^3(\theta) - 3\cos(\theta) + 2}{3}$
4	$\dfrac{3\theta - (3\sin(\theta) + 2\sin^3(\theta))\cos(\theta)}{8}$
5	$\dfrac{4\cos^3(\theta) - (3\sin^4(\theta) + 12)\cos(\theta) + 8}{15}$
>5	$\dfrac{d-1}{d} I_{d-2}(\theta) - \dfrac{\cos(\theta)\sin^{d-1}(\theta)}{d}$

the solution can be found by the recursive formula shown on the last row of the table.

As with linear correlation, there is usually a difference between random-Work and random-watermark false positive probabilities, insofar as there is usually a difference between the distributions from which Works and reference marks are drawn. The accuracy of the estimates obtained from Equation 7.30 depends on how close the distribution is to the radially symmetric ideal.

INVESTIGATION

An Evaluation of the Spherical Method

For the random-watermark false positive probability we can ensure that the distribution is radially symmetric by drawing the elements of watermark vectors independently from identical Gaussian distributions. However, for a random Work, the distribution is generally not radially symmetric. In this case, the false positive rate can become more predictable through the use of whitening.

This Investigation describes three experiments. In Experiment 1, we report on the accuracy of the spherical method for estimating the random-watermark false positive probability. Experiment 2 then measures the accuracy of the method for estimating the random-Work false positive probability. To improve the performance of this estimate, we introduce E_BLK_BLIND/D_WHITE_BLK_CC (System 10), which applies a whitening filter prior to applying the correlation coefficient detector. Experiment 3 measures the accuracy of the spherical method for estimating the random-Work false positive rate of this system.

Experiment 1

Figure 7.14 shows the measured and predicted random-watermark false positive probability using the D_BLK_CC watermark detector. Figure 7.14 plots the estimates obtained using Equation 7.30 against the results of the experiment shown in Figure 7.13. Figure 7.14 illustrates a near-perfect match between prediction and observation, even at high thresholds.

Experiment 2

Next, we look at the case of estimating the random-Work false positive probability using the spherical method. Of course, in the case of random-Work false positive probabilities, the distribution is unlikely to be radially symmetric.

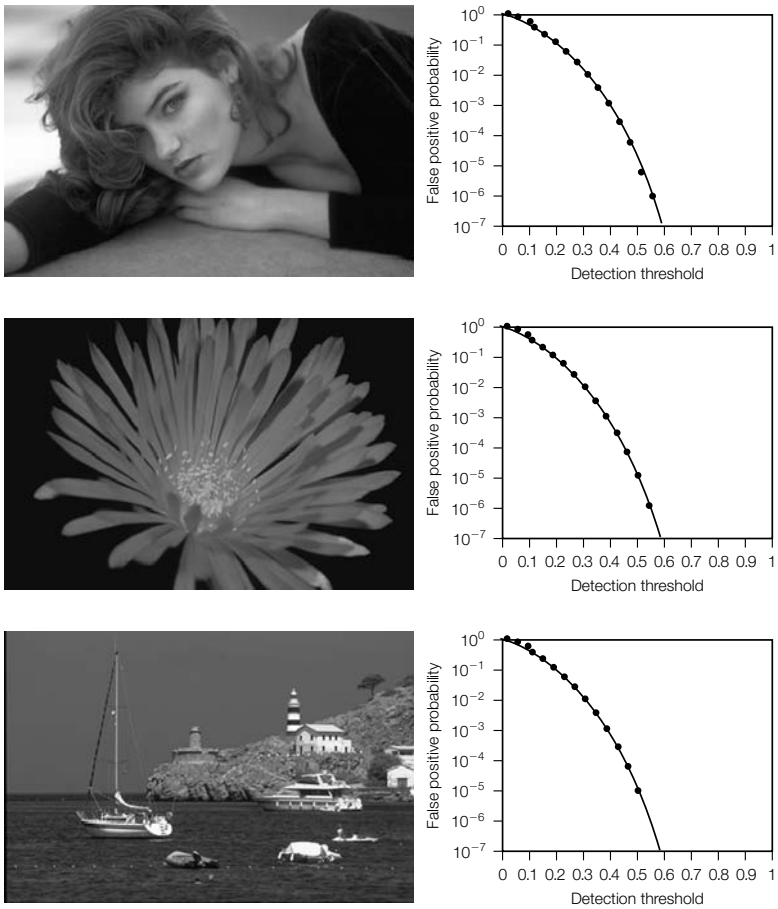


FIGURE 7.14

Random-watermark false positive rates for the correlation coefficient–based D_BLK_CC detection algorithm. Empirical results (points) are for 1,000,000 random reference marks compared against the three illustrated images, using the D_BLK_CC detector. Estimated probabilities (solid lines) are computed using the spherical method.

Figure 7.15 shows the random-Work false positive rates for three different reference marks using the D_BLK_CC detector. These were computed by testing 20,000 images.

Again, the points show the observed false positive rates and the solid lines show the prediction. Here we find that the observed rates are often quite different from the estimates, indicating that the distribution from which the summed 8×8 blocks are drawn is not radially symmetric.

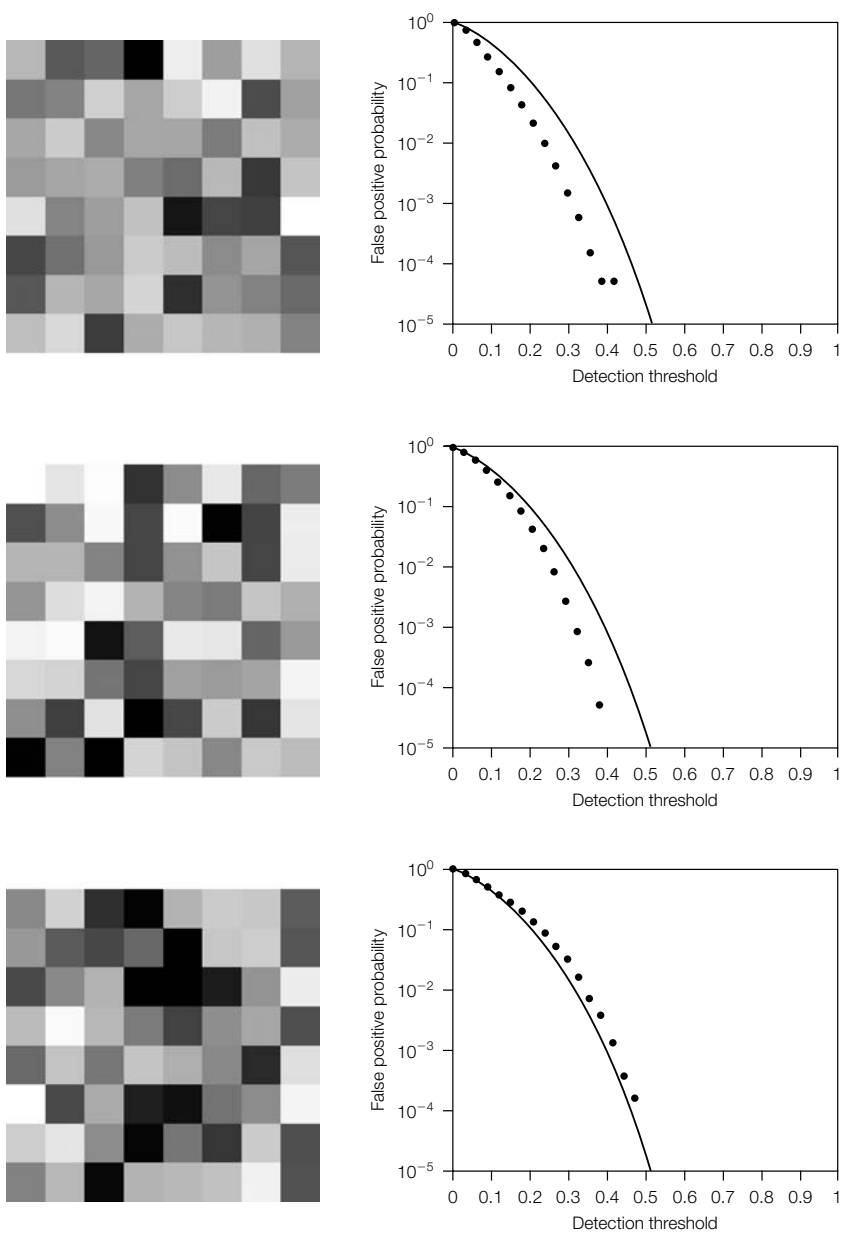


FIGURE 7.15

Random-Work false positive rates for D_BLK_CC detector. Results (points) are for 20,000 images from the Corel database compared against the three illustrated reference marks using the D_BLK_CC detector. Estimated probabilities (solid lines) are computed using the spherical method.

System 12: E_BLK_BLIND/D_WHITE_BLK_CC

We can rectify the problem to some extent by using a whitening filter, as described for linear correlation. This leads to the D_WHITE_BLK_CC detection method. This is a modification of the E_BLK_BLIND/D_BLK_CC watermarking system (System 3). The only difference is that the D_WHITE_BLK_CC detector applies the whitening filter derived for the D_WHITE detection algorithm (System 9).

Recall that the extraction process used in the E_BLK_BLIND/D_BLK_CC system divides the image into 8×8 blocks, and then averages them to obtain a single 8×8 block that serves as a 64-dimensional projected vector. If the pixels in the image are uncorrelated, the elements of the extracted vector will also be uncorrelated.

Of course, the pixels in most images are not uncorrelated, but we can roughly decorrelate them by applying the whitening filter employed in the D_WHITE detector (see Table 7.1). Because the summation is a linear process that preserves adjacency, we will obtain essentially the same results if we apply the filter after accumulation (with wraparound) as we would obtain by applying the filter before accumulation. Thus, the D_WHITE_BLK_CC detection algorithm begins by performing the averaging of the D_BLK_CC algorithm, to obtain an extracted 8×8 block, $\mathbf{v}$. It then applies the whitening filter to both the extracted block and the reference mark, $\mathbf{w_r}$, and computes the normalized correlation between them:

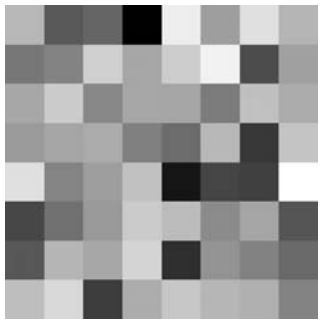
$$\begin{aligned} \mathbf{v}' &= \mathbf{f_{wh}} * \mathbf{v} - \overline{(\mathbf{f_{wh}} * \mathbf{v})} \\ \mathbf{w_r}' &= \mathbf{f_{wh}} * \mathbf{w_r} - \overline{(\mathbf{f_{wh}} * \mathbf{w_r})}. \end{aligned} \tag{7.31}$$

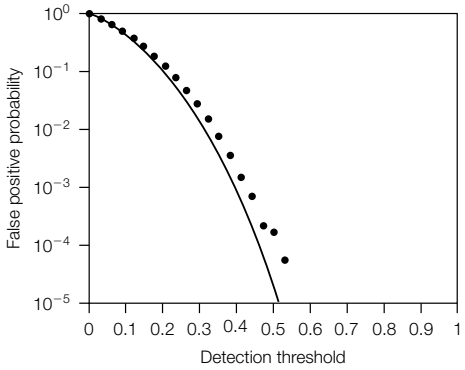
If the detection value, $z_{cc}(\mathbf{v}', \mathbf{w_r}')$, is above a threshold, τ_{wcc}, the detector reports that a 1 was embedded. If it is below the negative of the threshold, it reports that a 0 was embedded. Otherwise, it reports that there was no watermark detected.

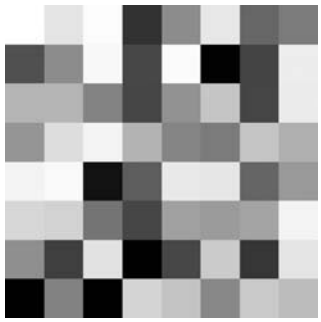
Because of the whitening filter, we expect the elements of $\mathbf{v}'$ to have low correlation with one another. Because of the accumulation, we expect each element of $\mathbf{v}'$ to be drawn from an approximately Gaussian distribution, since it is the sum of a large number of roughly uncorrelated values. Because we can assume that pixels in different locations are drawn from the same distribution, we expect the Gaussian distributions from which the elements of $\mathbf{v}'$ are drawn to be identical. This means that $\mathbf{v}'$ should be drawn from a distribution that is close to radially symmetric.

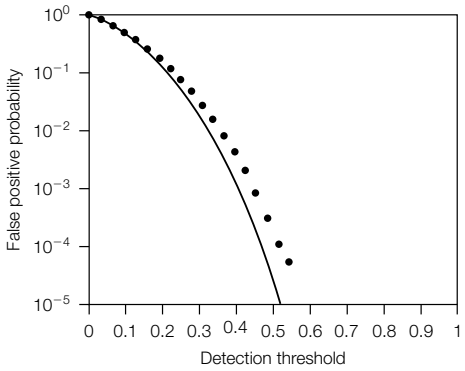
Experiment 3

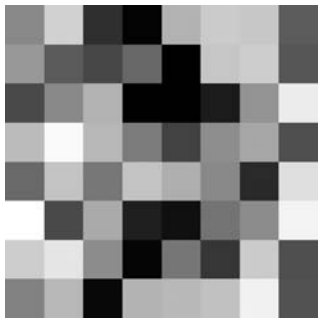
The D_WHITE_BLK_CC system leads to random-Work false positive behavior that is much more predictable than that for D_BLK_CC. Figure 7.16 shows the false positive results when we perform the same experiment as that shown in Figure 7.15. These results are closer to the false positive rates predicted by Equation 7.30.











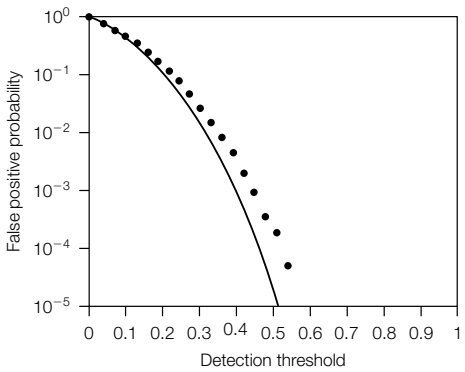


FIGURE 7.16

Random-Work false positive rates for D_WHITE_BLK_CC detection algorithm. Results are for 20,000 images from the Corel database compared against the three illustrated reference marks using the D_WHITE_BLK_CC detector.

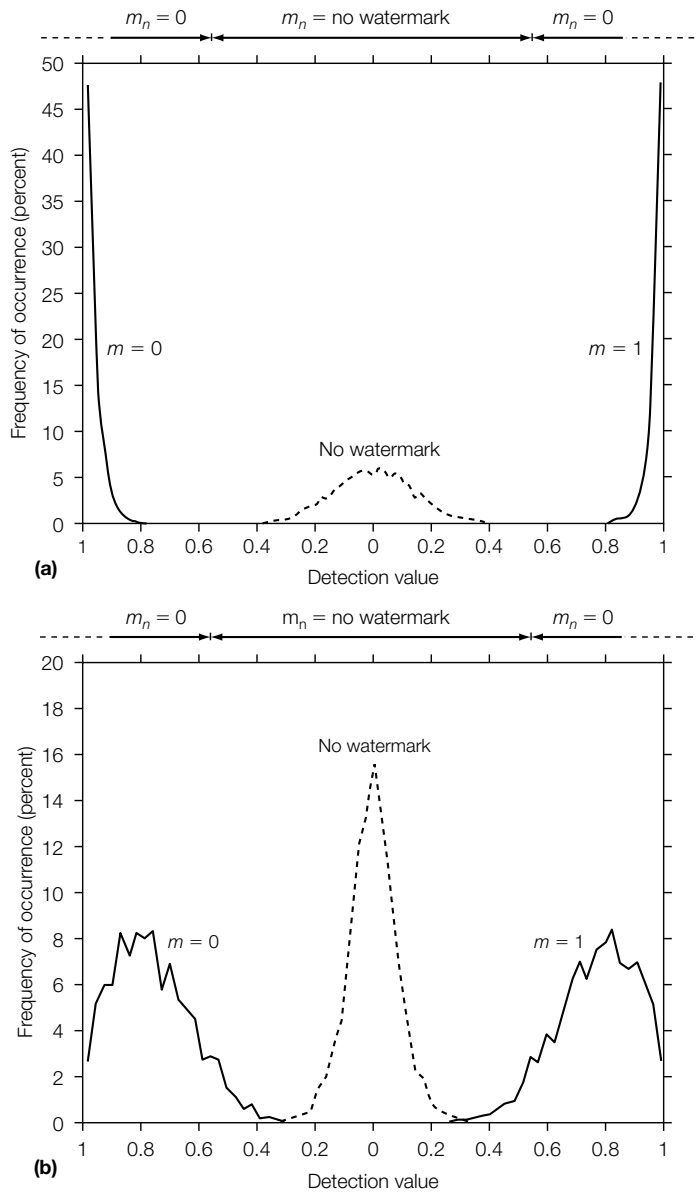


FIGURE 7.17

Comparison of correlation coefficient detection, with and without whitening, on watermarks embedded with the E_BLK_BLIND embedder. (a) Performance of D_WHITE_BLK_CC detection algorithm and (b) performance of D_BLK_CC detection algorithm, as reported in Figure 3.20.

Figure 7.17(a) shows the performance of the D_WHITE_BLK_CC detector when applied to images that were marked with the E_BLK_BLIND embedder. This should be compared against Figure 3.20, which is reproduced here as Figure 7.17(b). The comparison shows that applying the whitening filter significantly improves the separation between detection values from unwatermarked and watermarked images. This improvement is similar to the improvement obtained by using a whitening filter in a linear correlation system (Figure 7.10).

7.6.2 False Negative Analysis

It is difficult to make general claims about the false negative performance of a watermarking system when only its detection statistic is specified. We therefore constrain ourselves to estimating the false negative probability for a specific combination of embedder and detector, under the assumption that Works are not altered between embedding and detection. Specifically, we estimate the effectiveness of a system that employs a blind embedder and a normalized correlation detector, similar to the E_BLK_BLIND/D_BLK_CC watermarking system (System 3). The false negative analysis is based on the spherical method of analyzing false positives.

We begin by estimating the random-watermark false negative probability (i.e., the probability that a randomly selected watermark will fail to be embedded in a given Work). We assume that we have a fixed, extracted mark, $\mathbf{v_o}$, which was extracted from the Work in question, and we select a random reference mark, $\mathbf{w_r}$, from a radially symmetric distribution. We further assume that the reference mark is normalized to have unit magnitude. We then embed the reference mark with blind embedding, such that

$$\mathbf{v_w} = \mathbf{v_o} + \alpha \mathbf{w_r}, \tag{7.32}$$

where α is a constant scaling factor entered by the user. The probability of a false negative, P_{fn}, is the probability that

$$z_{\text{nc}}(\mathbf{v_w}, \mathbf{w_r}) < \tau_{\text{nc}}. \tag{7.33}$$

In Section B.3 of Appendix B, it is shown that the converse of P_{fn}—that is, the effectiveness (the probability of a *true positive*)—can be found by applying Equation 7.30 with a threshold higher than the detection threshold. This threshold depends on the magnitude of the vector extracted from the original Work and the embedding strength. The resulting formula for the probability of true positives, P_{tp}, is

$$P_{\text{tp}} = \frac{\int_0^{\phi} \sin^{N-2}(u)\, du}{2\int_0^{\pi 2} \sin^{N-2}(u)\, du}, \tag{7.34}$$

where

$$\phi = \cos^{-1}(\tau_{nc}) + \sin^{-1}\left(\frac{\alpha}{|\mathbf{v_o}|}\sqrt{1-\tau_{nc}^2}\right). \tag{7.35}$$

Here, $|\mathbf{v_o}|$ is the magnitude of the vector extracted from the original Work, α is the embedding strength, and τ_{nc} is the detection threshold. The probability of a false negative is just $P_{fn} = 1 - P_{tp}$.

The case of random-Work false negatives is more complex. We have a specific reference mark and wish to estimate the probability that a blind embedder, using a fixed value of α, will be unable to embed this mark in a randomly selected Work. Estimating this probability is more complicated than estimating the random-watermark probability because we cannot assume that the Work is normalized to fixed magnitude. This means that whether the watermark is successfully embedded depends on both the direction and length of the vector extracted from the random Work.

If we assume that random Works are drawn from a radially symmetric distribution, and we know the probability density function for their magnitudes, we can estimate the probability of successful embedding by finding the expected value of Equation 7.34. However, as we already know that random Works are not drawn from a radially symmetric distribution (at least, without whitening), we limit the following investigation to random-watermark false negative probabilities.

INVESTIGATION

Evaluation of the Spherical Method of Estimating False Negative Rates

We used the E_BLK_BLIND/D_BLK_CC (System 3) to evaluate the accuracy of the spherical method for predicting the false negative rate.

Experiment

We embedded one million randomly generated watermarks in each of three images using the E_BLK_BLIND embedder. The embedding strength was set to a low value of $\alpha = 0.1$, so that the false negative rate would be high enough to measure. We then used the D_BLK_CC detector to measure the resulting detection value, and compared it against several detection thresholds.

The results are shown in Figure 7.18. Here, we plot the embedding effectiveness (i.e., 1 minus the false negative rate) as a function of detection threshold. The points show the experimental data, and the curves show the prediction made using Equation 7.34. This clearly indicates that the predictions are quite accurate.

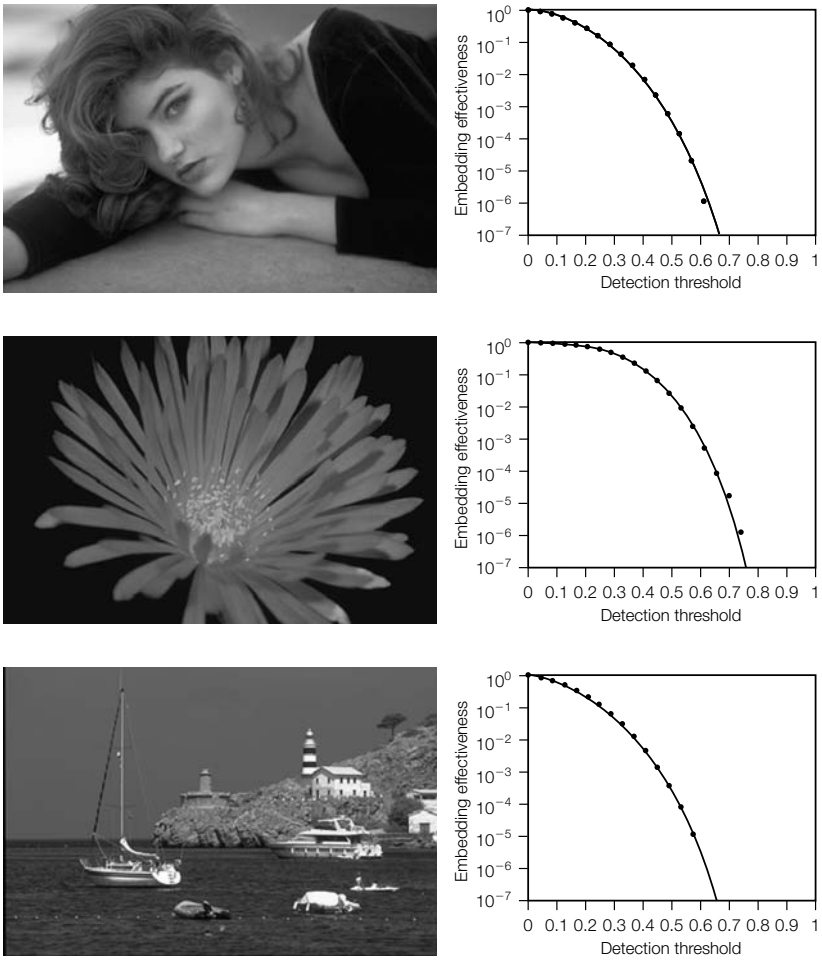


FIGURE 7.18

Results of effectiveness tests on the E_BLK_BLIND/D_BLK_CC system. The points show the fraction of watermarks successfully embedded in each image. The solid lines show the predicted effectiveness obtained from Equation 7.34.

7.7 SUMMARY

This chapter discussed the types of errors that occur in watermarking systems and how these errors can be modeled and measured. The main points of this chapter are the following.

- There are three main types of errors:
  - A *message error* occurs when the decoded message is not the same as the message that was embedded.
  - A *false positive* occurs when a watermark is detected in an unwatermarked Work.
  - A *false negative* occurs when a watermark is *not* detected in a watermarked Work.
- When messages are coded as binary strings, message error rates are often expressed as *bit error rates* (BER), which measure the frequency with which individual bits are incorrectly decoded.
- Error correction codes can be used to increase code separation, and thus decrease message error rates.
- The suitability of an error correction code depends on the watermarking application and the distortions that are anticipated.
- Various different types of false positive and false negative rates might be relevant:
  - *Random-watermark* false positives and negatives occur when a Work is fixed and the watermark is chosen at random. These are easy to analyze because the distribution of watermarks is determined by the design of the system. However, they are not necessarily relevant in most applications.
  - *Random-Work* false positives and negatives occur when the watermark is fixed and the Work is chosen at random. These are more difficult to analyze because they depend on the distribution of unwatermarked content.
- The tradeoff between false positive and false negative rates can be plotted with a *receiver operating characteristic* (ROC) curve.
- It is rarely possible to plot an ROC curve using only empirical data. One or both axes must be estimated with a model.
- Many watermarking systems are based on the assumption that the samples or pixels in unwatermarked content are uncorrelated. This assumption is generally incorrect.
- Application of a *whitening filter* can help decorrelate samples of a cover Work, and improve the performance of systems designed on this assumption. This was demonstrated for both linear correlation and normalized correlation systems.

- If a whitening filter is applied to signals for which it was not designed, it can *introduce* correlations and increase error rates. Thus, whitening filters must be used with caution.
- Two methods for analyzing the performance of normalized correlation detectors were presented:
  - The *approximate Gaussian* method models the distribution of detection values as a Gaussian distribution. This yields reasonable predictions at low detection thresholds, but overestimates false positive probabilities as the threshold increases toward 1.
  - The *spherical* method yields exact predictions of false positive probabilities under the assumption that random vectors are drawn from a radially symmetric distribution. It can also be used to predict random-watermark false negative probabilities for blind embedders.

CHAPTER 8

Using Perceptual Models

Watermarks are supposed to be imperceptible. This raises two important questions. How do we measure a watermark's perceptibility? And how can we embed a watermark in a Work such that it cannot be perceived? In this chapter, we describe the use of perceptual models to answer these questions.

In many applications, design tradeoffs prevent a watermark from being imperceptible in all conceivable Works, viewed or heard under all conditions by all observers. However, imperceptibility should not be viewed as a binary condition. A watermark may have a higher or lower level of perceptibility, meaning that there is a greater or lesser likelihood that a given observer will perceive it. Section 8.1 discusses some basic experimental methodologies for measuring this likelihood and introduces the idea of estimating perceptibility with automated perceptual models.

Any automated perceptual model must account for a variety of perceptual phenomena, the most important of which are briefly described in Section 8.2. Many models of the human auditory system (HAS) [33, 318] and the human visual system (HVS) [105, 274, 417, 429, 439] have been constructed. In Section 8.3, we describe simple models of human audition and vision.

Finally, Section 8.4 describes several ways a perceptual model can be incorporated into a watermarking system to control the perceptibility of watermarks. These range from using the model for simple adjustment of a global embedding strength to distorting the embedded mark so as to maximize detectability for a given perceptual impact. Example image watermarking techniques, using the HVS model of Section 8.3.1, are described and tested in this section.

8.1 EVALUATING PERCEPTUAL IMPACT OF WATERMARKS

Few, if any, watermarking systems produce watermarks that are perfectly imperceptible. However, the perceptibility of a given system's watermarks may be high or low compared against other watermarks or other types of processing, such

as compression. In this section, we address the question of how to measure that perceptibility, so that such comparisons can be made. We begin with a discussion of two types of perceptibility that can be of concern.

8.1.1 Fidelity and Quality

In the evaluation of signal-processing systems, there are two subtly different types of perceptibility that may be judged: *fidelity* and *quality*. Fidelity is a measure of the similarity between signals before and after processing. A high-fidelity reproduction is a reproduction that is very similar to the original. A low-fidelity reproduction is dissimilar, or distinguishable, from the original. Quality, on the other hand, is an absolute measure of appeal. A high-quality image or high-quality audio clip simply looks or sounds good. That is, it has no obvious processing artifacts. Both types of perceptibility are significant in evaluating watermarking systems.

To explore the difference between fidelity and quality, consider the example of video from a surveillance camera. The video is typically grayscale, low resolution, compressed, and generally considered to be of low *quality*. Now consider a watermarked version of this video that looks identical to the original. Clearly, this watermarked Work must also have low quality. However, because it is indistinguishable from the original, it has high *fidelity*.

It is also possible to have high quality and low fidelity. Consider a watermarking scheme that modifies the timbre of the baritone in a recording of a quartet. The watermarked Work may sound very good (i.e., have very high quality). In fact, depending on how the algorithm works, the change in the baritone's voice might actually be considered an *improvement* in quality. It would, however, be distinguishable from the original and therefore have a lower fidelity. In this case, a listener will be able to distinguish between the original and watermarked Works, but of the two, may not be able to identify the original.

For some watermarking applications, fidelity is the primary perceptual measure of concern. In these cases, the watermarked Work must be indistinguishable from the original. An artist may require this of a transaction watermark or a patient may require this of a watermark applied to his or her medical images. However, there are applications of watermarking for which quality, rather than fidelity, is the primary perceptual concern. After all, it is rare that the viewer/listener of a watermarked Work has access to the unwatermarked original for comparison. In these cases, the quality of a watermarked Work is required to be as high as that of the original. Thus, the important quantity is the *change* in quality due to the watermarking process.

Consideration must be given to the appropriate versions of the original and watermarked Work used in any evaluation of quality and fidelity. For example, consider a movie watermarked immediately after the telecine process (the process that digitizes the frames of the film). At this point, the image quality is extremely high. Fidelity and quality can be evaluated in the production studio,

directly after the watermark is applied to the video source, but this is not the content seen by consumers. Instead, we are often shown an NTSC version of the movie, which may have also undergone lossy MPEG-2 compression along part of the broadcast channel. Thus, it may be more appropriate for quality and fidelity evaluations to take place under normal viewing conditions (i.e., after the broadcast process and on a consumer television). The International Telecommunications Union (ITU), in its guidelines for the testing of television picture quality, specifies separate testing conditions for home and laboratory assessments [205].

The importance of including the transmission channel in the evaluation is that the transmission channel can have significant effects on the perceptibility of a watermark. In some cases, the noise introduced by the channel can act to hide the watermark noise, thus making the watermark less perceptible.[1] In other cases, however, the channel can increase the perceptibility of the watermark. For example, adaptive compression schemes may devote more bits to representation of the watermark component, thus reducing the number of bits devoted to the original content of the Work and thereby reducing the quality of the compressed watermarked Work.

8.1.2 Human Evaluation Measurement Techniques

Although the claim of imperceptibility is often made in the watermarking literature, rigorous perceptual quality and fidelity studies involving human observers are rare. Some claims of imperceptibility are based on automated evaluations, discussed in Section 8.1.3. However, many claims are based on a single observer's judgments on a small number of trials. These empirical data points are not sufficient for proper perceptual evaluation or comparison of watermarking algorithms. However, the two experimental techniques described next can provide statistically meaningful evaluations.

In studies that involve the judgment of human beings, it is important to recognize that visual and auditory sensitivities can vary significantly from individual to individual. These sensitivities also change over time in any one individual. Therefore, it is common that studies involving human evaluation use a large number of subjects and perform a large number of trials. The results of a study may be specific to the population from which the subjects are drawn (e.g., young adults, ages 18 to 25).

Perhaps the most important point to note here is that the experiments are statistical in nature. Different observers will behave differently. One observer might claim to see a difference in a pair of images, whereas another observer may

[1] The concept of one signal (in this case the transmission noise) reducing the perceptibility of another (in this case hiding the watermark noise) is called *masking* and is discussed later in this chapter.

not. Sometimes these discrepancies are random. Other times, the discrepancies reflect the very different perceptual abilities of observers. In fact, it is well known that a small percentage of people have extremely acute vision or hearing. In the music industry these people are often referred to as *golden ears* and in the movie industry as *golden eyes,* and they are commonly employed at quality control points in the production process.

In psychophysics studies, a level of distortion that can be perceived in 50% of experimental trials is often referred to as a *just noticeable difference,* or *JND*. Even though some golden eyes or golden ears observers might be able to perceive a fidelity difference of less than one JND, this difference is typically considered the minimum that is generally perceptible. JNDs are sometimes employed as a unit for measuring the distance between two stimuli. However, while the notion of *one* JND has a fairly consistent definition (perceptible 50% of the time), multiple JNDs can be defined in various ways. Watson, in the work described in Section 8.2, defines JNDs as linear multiples of a noise pattern that produces one JND of distortion [438].

A classical experimental paradigm for measuring perceptual phenomena is the *two alternative, forced choice* (2AFC) [168]. In this procedure, observers are asked to give one of two alternative responses to each of several trial stimuli. For example, to test the quality impact of a watermarking algorithm, each trial of the experiment might present the observer with two versions of one Work. One version of the Work would be the original; the other would be watermarked. The observer, unaware of the differences between the Works, must decide which one has higher quality. In the case where no difference in quality can be perceived, we expect the responses to be random. Random choices suggest that observers are unable to identify one selection as being consistently better quality than the other. Thus, 50% correct answers correspond to zero JND, while 75% correct correspond to one JND.

The 2AFC technique can also be used to measure fidelity. Consider an experiment in which the observer is presented with three Works. One is labeled as the original. Of the other two, one is an exact copy of the original and the other is the watermarked version. The subject must choose which of the two latter Works is identical to the original (see Figure 8.1). Again, the results are tabulated and examined statistically. Any bias in the data represents the fact that the observers could distinguish between the original and watermarked Works, and serves as a measure of the fidelity of the watermarking process.

An interesting variation on these tests is to vary the strength of the watermark during the experiment. Typically, an S-type curve is expected when the probability of correct answers is plotted against the watermark strength. Clearly, when the strength is very low, the probability that the watermark will be correctly identified is 50% (i.e., random chance). When the watermark strength is very high, the probability of a correct answer will be 100%. In the intermediate region between these two extremes, the nonzero probability of discernment indicates that some observers are able to perceive the watermark

(a) Original

(b)

(c)

FIGURE 8.1

A two alternative, forced choice experiment studying image fidelity. The observer is asked to choose whether image (b) or image (c) is the same as image (a), labeled "Original."

Table 8.1 Quality and impairment scale as defined in ITU-R Rec. 500.

Five-Grade Scale			
Quality		**Impairment**	
5	Excellent	5	Imperceptible
4	Good	4	Perceptible, but not annoying
3	Fair	3	Slightly annoying
2	Poor	2	Annoying
1	Bad	1	Very annoying

in some Works. The perceptual goal of a watermarking scheme is to keep these probabilities under some application-specific threshold.

A second, more general experimental paradigm for measuring quality allows the observers more latitude in their choice of responses. Rather than selecting one of two Works as the "better," observers are asked to rate the quality of a Work, sometimes with reference to a second Work. For example, the ITU-R Rec. 500 quality rating scale specifies a quality scale and an impairment scale that can be used for judging the quality of television pictures [205]. These scales, summarized in Table 8.1, have been suggested for use in the evaluation of image watermarking quality [246].

8.1.3 Automated Evaluation

The experimental techniques previously outlined can provide very accurate information about the fidelity of watermarked content. However, they can be very expensive and are not easily repeated. An alternative approach is the use of an algorithmic quality measure based on a perceptual model. The goal of a perceptual model is to predict an observer's response. The immediate advantages of such a system are that it is faster and cheaper to implement and, assuming the use of a deterministic model, the evaluation is repeatable, so that different methods can be directly compared. A number of perceptual models are available for judging the fidelity of watermarked Works. One in particular [417] has been proposed for use in normalizing various image watermarking techniques prior to a benchmark evaluation of robustness [247].

In practice it is very difficult to accurately predict the human judgment of perceptual fidelity, and even more difficult to predict human judgment of perceptual quality. Most of the models currently available measure fidelity,

evaluating the perceptible difference between two signals. Recall, however, that high fidelity implies little change in quality. Two Works that are truly indistinguishable necessarily have the same quality.

Ideally, a perceptual model intended for automated fidelity tests should predict the results of tests performed with human observers. However, for purposes of comparing the fidelity of different watermarking algorithms, it is sufficient for the model to provide a value that is monotonically related to the results of human tests. That is, it is not necessary for the model to predict the exact results of tests performed with human observers, but simply to predict the relative performance of different algorithms in those tests. Thus, it is sufficient for the model to produce a measure of the perceptual distances between watermarked and unwatermarked Works without calibrating those distances in terms of expected test results.

When we refer to a "perceptual model," we mean more precisely a function, $D(\mathbf{c_o}, \mathbf{c_w})$, that gives a measure of the distance between an original Work, $\mathbf{c_o}$, and a watermarked Work, $\mathbf{c_w}$. One of the simplest distance functions is the mean square error (MSE) function discussed in Chapter 3. This is defined as

$$D_{\text{mse}}(\mathbf{c_o}, \mathbf{c_w}) = \frac{1}{N}\sum_{i}^{N}(\mathbf{c_w}[i] - \mathbf{c_o}[i])^2. \tag{8.1}$$

Although MSE is often used as a rough test of a watermarking system's fidelity impact, it is known to provide a poor estimate of the true fidelity [162, 208].

A perceptual model such as MSE can err by either underestimating or overestimating the perceptibility of the difference between two Works. For example, suppose we interpret MSE as indicating the visual impact of adding white noise. Then, as shown in Figure 8.2, MSE generally underestimates the perceptual difference when that difference is a low-frequency signal. The figure shows an image and two modified versions of it. The top modified image was produced by adding white noise to the original. The bottom image was produced by adding low-pass-filtered noise. MSE, as a perceptual model, suggests that these two images should be perceptually equivalent, in that the two images have the same MSE compared to the original. However, clearly the low-pass signal is more visible.

On the other hand, Figure 8.3 shows a case in which MSE overestimates perceptual distance. Here, the second modified image was produced by simply shifting the original down and to the right. Although the original and the shifted images are almost indistinguishable, the MSE between them is very large. For comparison, the top modified image was produced by adding enough white noise to get the same MSE. Clearly, MSE is wrong in predicting that these two versions of the original should be perceptually equivalent.

Care must be taken in choosing the perceptual model used for fidelity evaluations and in interpreting the results of such a model. The issues involved in model selection include the tradeoff between accuracy and computational cost, and the fact that the accuracy of any particular model may be limited to

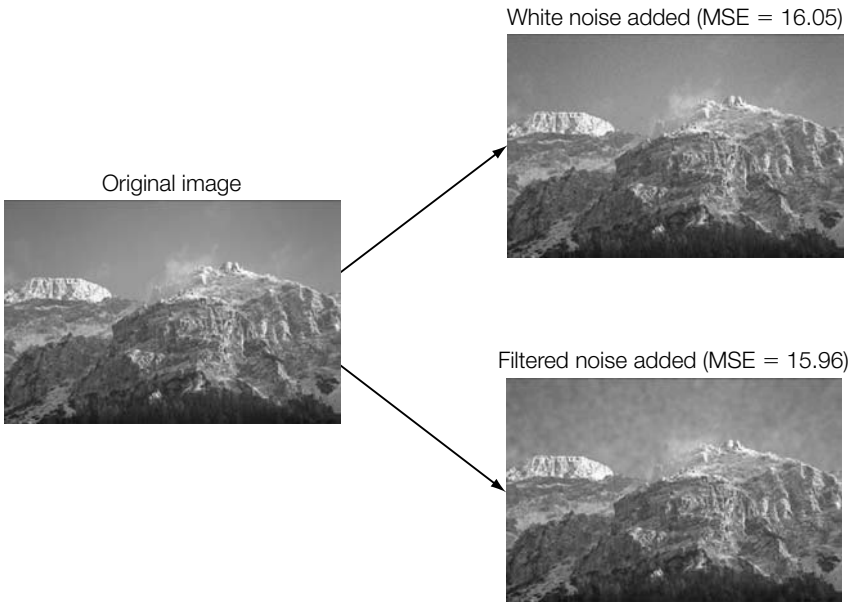


FIGURE 8.2

Illustration of a case in which MSE underestimates perceptual distance.

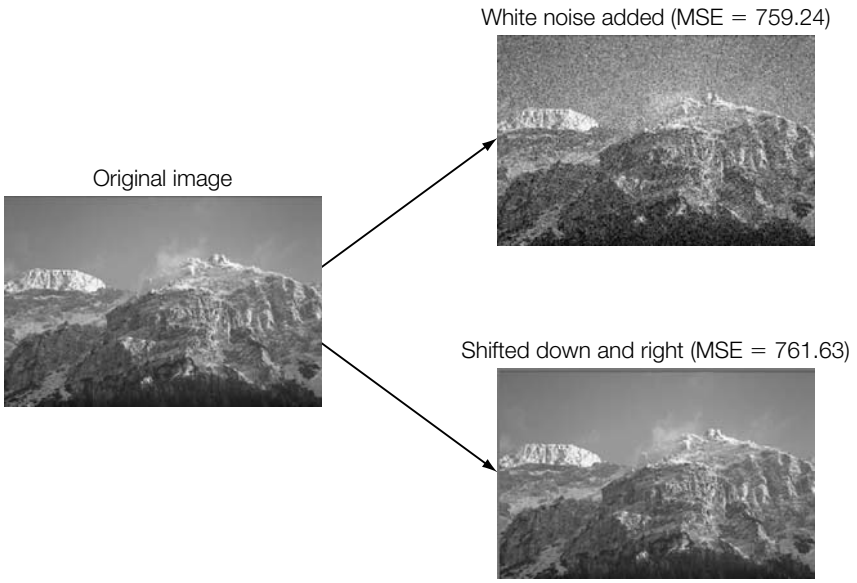


FIGURE 8.3

Illustration of a case in which MSE overestimates perceptual distance.

a class of distortions. For example, like MSE, many visual models overestimate the perceptual impact of slight geometric transformations (rotation, scaling, translation, skew, and so on). A watermarking algorithm that introduces geometric transformations, such as Maes and van Overveld [281], is bound to perform poorly when measured with such a model, even if the changes are completely imperceptible.

8.2 GENERAL FORM OF A PERCEPTUAL MODEL

The problem with the MSE measure is that it treats changes in all terms of the Work equally. However, human perception mechanisms are not uniform. For example, the HAS responds differently depending on the frequency and loudness of the input. Similarly, the response of the HVS varies with the spatial frequency, brightness, and color of its input. This suggests that all components of a watermark may not be equally perceptible.

Perceptual variations can be measured, and models constructed to account for them. A perceptual model generally attempts to account for three basic types of phenomena: *sensitivity, masking,* and *pooling*. In this section, we briefly explain what is meant by these three terms.

8.2.1 Sensitivity

Sensitivity refers to the ear's or eye's response to direct stimuli. In experiments designed to measure sensitivity, observers are presented with isolated stimuli and their perception of these stimuli is tested. For example, it is common to measure the minimum sound intensity required to hear a particular frequency and to repeat this over a range of frequencies. Although there are many different aspects of a signal to which the eye or ear is sensitive, the primary stimuli or characteristics measured are frequency and loudness (or brightness). Color and orientation are also significant in image and video data.

Frequency Sensitivity

The responses of both the HAS and HVS to an input signal are frequency dependent. In hearing, variations in frequency are perceived as different tones. Figure 8.4 shows one model of the ear's sensitivity as a function of frequency [407]. The graph gives the minimum audible sound level, which is the reciprocal of sensitivity, for each frequency. This curve indicates that the ear is most sensitive to frequencies around 3 kHz and sensitivity declines at very low (20 Hz) and very high (20 kHz) frequencies.

In vision, there are three forms of frequency response. One is to *spatial* frequencies, a second is to *spectral* frequencies, and the third is to *temporal* frequencies.

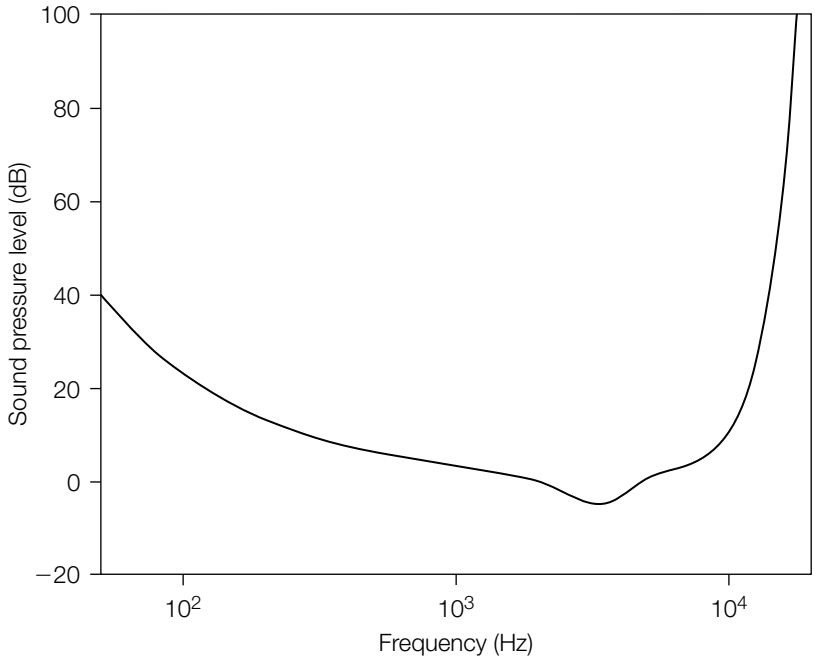


FIGURE 8.4

Frequency response of the human ear based on the Terhardt model [407]. This curve represents the absolute threshold, or quiet threshold, for an average young listener with acute hearing.

Spatial frequencies are perceived as patterns or textures. The spatial frequency response is usually described by the sensitivity to luminance contrast (i.e., changes in luminance) as a function of spatial frequency. This is called the *contrast sensitivity function* (CSF), one model of which is illustrated in Figure 8.5 [282]. The figure clearly implies that we are most sensitive to luminance differences at midrange frequencies and that our sensitivity decreases at lower and higher frequencies. This is analogous to the human auditory system.

Two-dimensional spatial frequency patterns can be represented by their magnitude and orientation. It has been shown that the sensitivity of the eye is not only dependent on frequencies of different patterns but on their orientations [61, 62, 403]. In particular, the eye is most sensitive to vertical and horizontal lines and edges in an image and is least sensitive to lines and edges with a 45-degree orientation.

Spectral frequencies are perceived as colors. The lowest level of color vision consists of three separate color systems [87]. The normal responses of each of

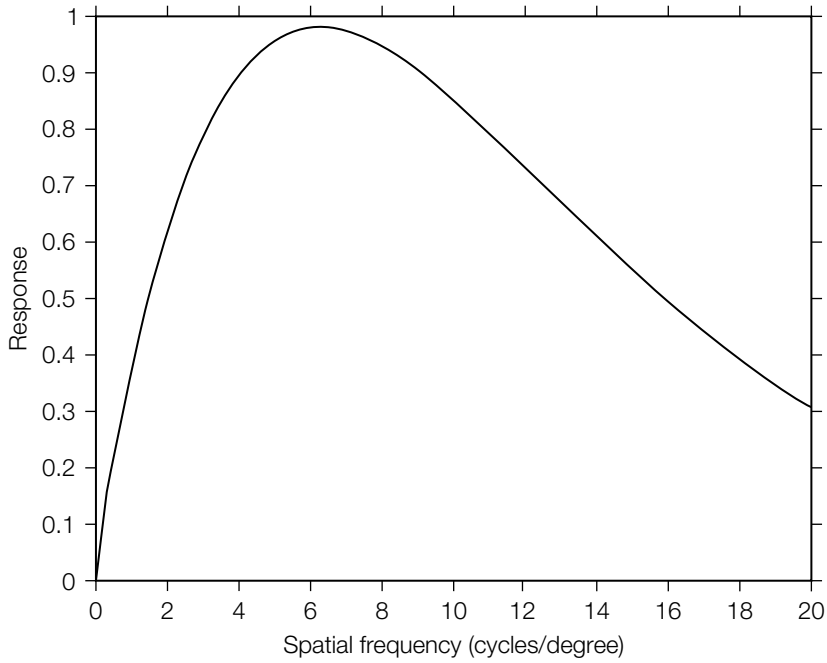


FIGURE 8.5

Contrast sensitivity function based on the model of Mannos and Sakrison [282]. This curve represents the reciprocal of the minimum spatial sinusoidal amplitude, at each frequency, at which the variation is perceived.

these systems are shown in Figure 8.6. The low-frequency response, often called the blue channel, is shown to be significantly lower than the other two channels. For this reason, several color watermarking systems place a large proportion of the watermark signal in the blue channel of an RGB image [180, 245, 360].

Temporal frequencies are perceived as motion or flicker. Figure 8.7 shows the result of an experiment measuring the eye's response to various temporal frequencies [223]. The result shows that sensitivity falls off very rapidly for frequencies above 30 Hz. This is why television and cinema frame rates do not exceed 60 frames per second.

Loudness/Brightness Sensitivity

A number of studies of human hearing have measured the minimum intensity change detectable as a function of intensity level [59]. Although the experimental approaches can differ, the general result is that we are able to discern smaller changes when the average intensity is louder (i.e., the human ear is more sensitive to changes in louder signals than in quieter signals).

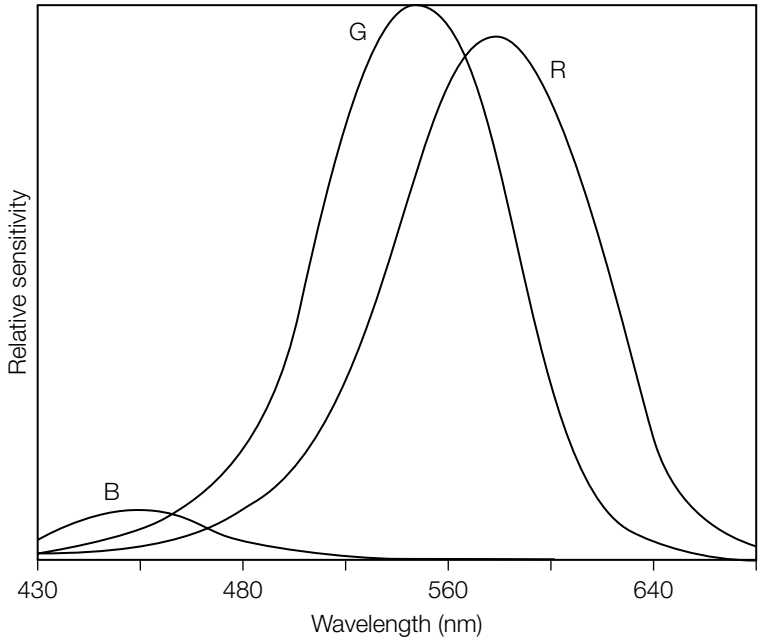


FIGURE 8.6

The three color systems in normal human color vision.

The opposite is true of the eye, which is less sensitive to brighter signals. Brightness sensitivity is nonlinear [87] and has been modeled with a log relationship [171, 388], cube root relationship [282], and with more complicated models [105]. It is a common practice in image processing to compensate for this nonlinearity prior to the application of linear processes.

8.2.2 **Masking**

Context affects perception. Thus, although we might be able to hear an isolated tone at some particular sound intensity, this tone may become completely inaudible if a second tone at a nearby frequency is louder. Similarly, a texture that is easy to see in isolation might be difficult to see when added to a highly textured image. That is, the presence of one signal can hide or mask the presence of another signal. Masking is a measure of an observer's response to one stimulus when a second "masking" stimulus is also present.

Figure 8.8 shows an original image and the result of adding a uniform noise pattern to it. Although the noise is uniform, its visibility within the image is very nonuniform and is clearly dependent on the local image structure. In the relatively flat, homogeneous regions of the sky, the noise is clearly visible. In

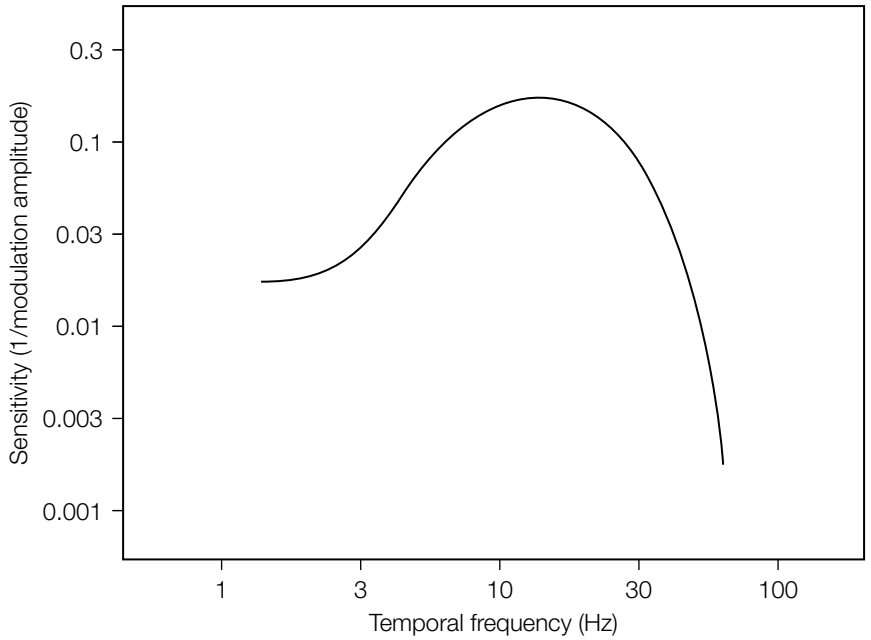


FIGURE 8.7

This curve shows the measured sensitivity of the HVS to temporal sinusoidal variation. The data represents one of a family of curves given in Kelly [223]. The sensitivity is the reciprocal of the minimum temporal sinusoidal amplitude, at each frequency, at which the variation is perceived. For a more detailed discussion of this experiment, see Kelly [223] or Cornsweet [87].

comparison, in the highly textured mountains, the noise is almost completely invisible.

There are many masking phenomena. In vision, two principal cases are frequency masking, in which the presence of one frequency masks the perception of another, and brightness masking, in which the local brightness masks contrast changes. In audio, there exist similar frequency and loudness masking conditions. In addition, temporal masking also occurs in which the perception of a sound may be masked by a previous sound or even by a future sound.

8.2.3 Pooling

Sensitivity and masking models can be used to provide an estimate of the perceptibility of a change in a particular characteristic (e.g., a single frequency). However, if multiple frequencies are changed rather than just one, we need to know how to combine the sensitivity and masking information for each frequency. In a model of perceptual distance, combining the perceptibilities of separate distortions gives a single estimate for the overall change

(a) Original

(b) Watermarked

FIGURE 8.8

A low-frequency watermark pattern has been embedded into an image (a) by the E_BLIND embedding algorithm, resulting in a watermarked image (b). The watermark is plainly visible in the relatively flat regions of the image (sky) but is well masked in the textured areas (mountains).

in the Work. This is known as *pooling*. It is common to apply a formula of the form

$$D(\mathbf{c_o}, \mathbf{c_w}) = \left(\sum_i |\mathbf{d}[i]|^p \right)^{\frac{1}{p}}, \tag{8.2}$$

where $\mathbf{d}[i]$ is an estimate of the likelihood that an observer will notice the difference between $\mathbf{c_o}$ and $\mathbf{c_w}$ in an individual parameter, such as a temporal sample, spatial pixel, or Fourier frequency coefficient. Equation 8.2 is often referred to as a *Minkowski summation* or an L_p*-norm*. In the case of audio, a linear summation, with $p = 1$, may be appropriate, whereas for images, a value of $p = 4$ is more typical.

8.3 TWO EXAMPLES OF PERCEPTUAL MODELS

Using the ideas of sensitivity, masking, and pooling, we now describe two perceptual models. The first is due to Watson [438] and is a model for measuring visual fidelity. The second is an audio model that is a simplification of the MPEG-1 Layer 1 lossy audio compression standard [213].

Watson describes a perceptual model that tries to estimate the number of JNDs between images. This model is far better than MSE at estimating the perceptual effects of noise added to images. For the two noisy images of Figure 8.2, it gives distances of 38.8 and 158.8, respectively, thus reflecting the difference in perceptual fidelity. However, this model still relies on proper synchronization and overestimates the effect of shifts. It also underestimates the effect of certain blocking artifacts.

The MPEG-1 Layer 1 model does not try to estimate the number of JNDs between audio signals. Instead, it tries to identify those frequency components that are perceptually insignificant (i.e., the coefficients that are below the threshold of hearing). This model can be used in watermarking to identify the components that may be replaced with watermark data.

Although the two models measure different perceptual properties, both models have been used as the basis for watermarking algorithms [46, 160, 335, 397]. Later in this chapter, we use Watson's model as part of a number of investigations.

8.3.1 Watson's DCT-Based Visual Model

This model estimates the perceptibility of changes in individual terms of an image's block DCT (explained in the following), and then pools those estimates into a single estimate of perceptual distance, $D_{\text{wat}}(\mathbf{c_o}, \mathbf{c_w})$, where $\mathbf{c_o}$ is an original image and $\mathbf{c_w}$ is a distorted version of $\mathbf{c_o}$. Although this model, being block based, is not ideal for watermarking, we describe it here because it serves as a simple illustration of the ideas previously discussed.

Perceptual models can be based on a variety of signal representations. Watson's model uses the *block DCT transform,* which proceeds by first dividing the image into disjoint 8×8 blocks of pixels. If our image is denoted by $\mathbf{c}$, we denote the i, jth pixel in block number k by $\mathbf{c}[i, j, k]$, $0 \le i, j \le 7$. Each of these blocks is then transformed into the DCT domain, resulting in the block DCT of the image, $\mathbf{C}$. By $\mathbf{C}[i, j, k]$, $0 \le i, j \le 7$, we denote one term of the DCT of the kth block. $\mathbf{C}[0, 0, k]$ is the DC term (i.e., the mean pixel intensity in the block). This transform generally concentrates the image energy into the low-frequency coefficients of each block.

Watson chose to base his model on the block DCT domain because he intended it for use in JPEG image compression. JPEG compresses images by converting them to the block DCT domain and quantizing the resulting terms according to frequency-dependent step sizes. Using Watson's model to estimate perceptibility of the resulting quantization noise, it is possible to adapt the quantization step sizes to the specific characteristics of each image. However, we intend to use the model for evaluating and controlling watermark embedding algorithms. Therefore, we describe it here without further reference to JPEG.

Watson's model consists of a sensitivity function, two masking components based on luminance and contrast masking, and a pooling component.

Sensitivity

The model defines a frequency sensitivity table, $\mathbf{t}$. Each table entry, $\mathbf{t}[i, j]$, is approximately the smallest magnitude of the corresponding DCT coefficient in a block that is discernible in the absence of any masking noise (i.e., the amount of change in that coefficient that produces one JND). Thus, a smaller value indicates that the eye is more sensitive to this frequency. This sensitivity table is a function of a number of parameters, including the image resolution and the distance of an observer to the image. It is derived in Ahumada and Peterson [14]. In the implementation of this perceptual model for the examples in this chapter, a set of parameter values has been chosen. The resulting frequency sensitivity table is shown as Table 8.2.

Luminance Masking

Luminance adaptation refers to the fact that a DCT coefficient can be changed by a larger amount before being noticed if the average intensity of the 8×8 block is brighter. To account for this, Watson's model adjusts the sensitivity table, $\mathbf{t}[i, j]$, for each block, k, according to the block's DC term. The luminance-masked threshold, $\mathbf{t_L}[i, j, k]$, is given by

$$\mathbf{t_L}[i, j, k] = \mathbf{t}[i, j](\mathbf{C_o}[0, 0, k] / C_{0,0})^{a_T}, \tag{8.3}$$

where a_T is a constant with a suggested value of 0.649, $\mathbf{C_o}[0, 0, k]$ is the DC coefficient of the kth block in the original image, and $C_{0,0}$ is the average of the

Table 8.2 DCT frequency sensitivity table.

1.40	1.01	1.16	1.66	2.40	3.43	4.79	6.56
1.01	1.45	1.32	1.52	2.00	2.71	3.67	4.93
1.16	1.32	2.24	2.59	2.98	3.64	4.60	5.88
1.66	1.52	2.59	3.77	4.55	5.30	6.28	7.60
2.40	2.00	2.98	4.55	6.15	7.46	8.71	10.17
3.43	2.71	3.64	5.30	7.46	9.62	11.58	13.51
4.79	3.67	4.60	6.28	8.71	11.58	14.50	17.29
6.56	4.93	5.88	7.60	10.17	13.51	17.29	21.15

DC coefficients in the image. Alternatively, $C_{0,0}$ may be set to a constant value representing the expected intensity of images.

Equation 8.3 indicates that brighter regions of an image will be able to absorb larger changes without becoming noticeable. This is illustrated in Figure 8.9, which shows the luminance-masking values for the image of Figure 8.8(a).

Contrast Masking

The luminance-masked threshold, $\mathbf{t_L}[i, j, k]$, is subsequently affected by contrast masking. Contrast masking (i.e., the reduction in visibility of a change in one frequency due to the energy present in that frequency) results in a masking threshold, $\mathbf{s}[i, j, k]$, given by

$$\mathbf{s}[i,j,k] = \max\left\{\mathbf{t_L}[i,j,k], |\mathbf{C_o}[i,j,k]|^{w[i,j]}\mathbf{t_L}[i,j,k]^{1-w[i,j]}\right\}, \tag{8.4}$$

where $w[i, j]$ is a constant between 0 and 1 and may be different for each frequency coefficient. Watson uses a value of $w[i, j] = 0.7$ for all i, j. The final thresholds, $\mathbf{s}[i, j, k]$, estimate the amounts by which individual terms of the block DCT may be changed before resulting in one JND. We refer to these thresholds as *slacks*. Figure 8.10 shows the slacks computed for Figure 8.8(a).

Pooling

To compare an original image, $\mathbf{c_o}$, and a distorted image, $\mathbf{c_w}$, we first compute the differences between corresponding DCT coefficients, $\mathbf{e}[i, j, k] = \mathbf{C_w}[i, j, k] - \mathbf{C_o}[i, j, k]$. We then scale these differences by their respective slacks, $\mathbf{s}[i, j, k]$, to obtain the perceptible distance, $\mathbf{d}[i, j, k]$, in each term:

$$\mathbf{d}[i, j, k] = \frac{\mathbf{e}[i, j, k]}{\mathbf{s}[i, j, k]}. \tag{8.5}$$

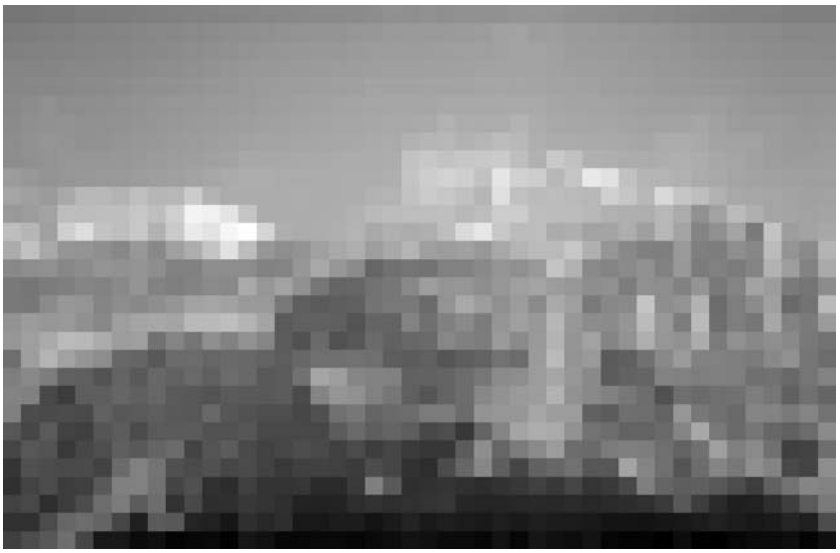


FIGURE 8.9

Relative luminance-masking thresholds for Figure 8.8(a). Bright areas indicate blocks with high luminance-masking values.

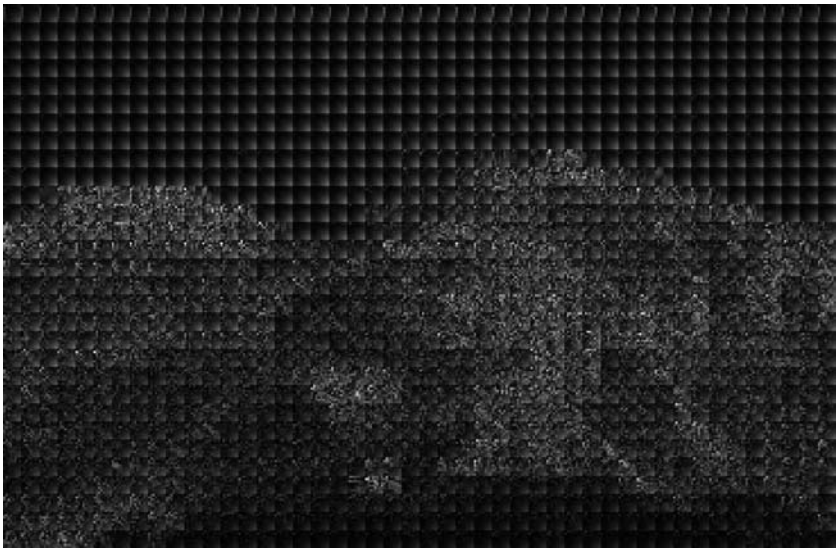


FIGURE 8.10

Relative masking thresholds, or slacks, for Figure 8.8(a). The brightness of each pixel here is proportional to the slack of the corresponding block-DCT coefficient.

Thus, $\mathbf{d}[i, j, k]$ measures the error in the i, jth frequency of block k as a fraction or multiple of one JND.

The individual errors computed in Equation 8.5 must be combined or pooled into a single perceptual distance, $D_{\text{wat}}(\mathbf{c_o}, \mathbf{c_w})$. Watson employs two forms of pooling. The first combines the errors across blocks, and the second combines the errors over the different frequencies within blocks. However, his suggested exponent for the L_p-norm is the same for both classes of errors. Therefore, the two forms of pooling can be combined into the single equation

$$D_{\text{wat}}(\mathbf{c_o}, \mathbf{c_w}) = \left(\sum_{i,j,k} |\mathbf{d}[i, j, k]|^p \right)^{\frac{1}{p}}, \tag{8.6}$$

where Watson recommends a value of $p = 4$.

8.3.2 A Perceptual Model for Audio

The audio model presented here, which is used for watermarking in [160, 397], provides a measure of the threshold of hearing in the presence of a given audio signal. For each frequency, the model estimates the minimum sound energy needed to be perceived. These estimates can be used to predict whether or not a given change in a given frequency will be audible.

To obtain the thresholds of hearing, we first partition the signal into overlapping *frames* or *windows*. This is usually accomplished using a Hamming or Hanning window [176] as a filtering step. Each audio frame, $\mathbf{c}[t]$, is then analyzed independently in the Fourier domain, applying sensitivity and frequency-masking functions to obtain an estimate, $\mathbf{t}[f]$, of the minimum sound energy needed for an observer to discern a tone at each frequency, f. Although in [160, 397] no attempt is made to pool the results into a single measure of overall perceptibility, we include a brief discussion of how pooling might be performed.

Sensitivity

Sound pressure level (SPL) measures the intensity of sound, in decibels (dB), relative to a reference intensity of 20 μ Pascals,[2] as

$$SPL = 20 \log_{10} \frac{p}{20}, \tag{8.7}$$

where p is the sound pressure of the stimulus in Pascals. The *absolute threshold of hearing* is defined in db SPL as the minimum intensity needed in a pure tone for it to be detected by a human in a noiseless environment. The absolute

[2] One Pascal is equivalent to one Newton per square meter (N/m^2).

threshold of hearing was previously illustrated in Figure 8.4 and can be modeled by the function

$$T_q(f) = 3.64\left(\frac{f}{1{,}000}\right)^{-0.8} - 6.5\exp^{-0.6(f/1{,}000-3.3)^2} + 10^{-3}\left(\frac{f}{1{,}000}\right)^{4}. \qquad (8.8)$$

The threshold, $T_q(f)$, is approximately the magnitude of the smallest change discernible in a quiet environment.

Masking

The ear can be modeled as a series of overlapping bandpass filters. These filters are characterized by their *critical bandwidth*. The critical bandwidth is characterized as follows. Consider a narrowband noise source that we perceive with a certain loudness. If the bandwidth of the noise source is increased, the perceived loudness will remain constant until the bandwidth exceeds the critical bandwidth, at which point we perceive an increase in loudness. The critical bandwidth varies with frequency measured in hertz. It is common to model the ear by a series of discrete critical bands, which we denote by z, where $1 \le z \le Z_t$, and Z_t is the total number of critical bands. These discrete critical bands are tabulated in Table 8.3.

The threshold in each critical band is a function of the apparent energy in that band and whether the frame sounds like noise or a tone. The apparent energy in each band depends on the real energy in a neighborhood of bands. Thus, to determine the masking threshold in each frequency, we must

Table 8.3 Critical band frequencies, taken from [318].

z	Center freq. in Hertz	Bandwidth L[z]–H[z]	z	Center freq. in Hertz	Bandwidth L[z]–H[z]	z	Center freq. in Hertz	Bandwidth L[z]–H[z]
1	50	–100	10	1,175	1,080–1,270	19	4,800	4,400–5,300
2	150	100–200	11	1,370	1,270–1,480	20	5,800	5,300–6,400
3	250	200–300	12	1,600	1,480–1,720	21	7,000	6,400–7,700
4	350	300–400	13	1,850	1,720–2,000	22	8,500	7,700–9,500
5	450	400–510	14	2,150	2,000–2,320	23	10,500	9,500–12,000
6	570	510–630	15	2,500	2,320–2,700	24	13,500	12,000–15,500
7	700	630–770	16	2,900	2,700–3,150	25	19,500	15,500–
8	840	770–920	17	3,400	3,150–3,700			
9	1,000	920–1,080	18	4,000	3,700–4,400			

1. Determine the energy in each critical band.
2. Determine the *apparent* energy in each critical band due to the spread of energy from one critical band to another.
3. Determine whether the current audio frame is noiselike or tonelike.
4. Determine the masking threshold.

The energy present in each critical band, $\mathbf{B}[z]$, is given by

$$\mathbf{B}[z] = \sum_{f=\mathbf{L}[z]}^{\mathbf{H}[z]} |\mathbf{C}[f]|, \tag{8.9}$$

where $\mathbf{L}[z]$ and $\mathbf{H}[z]$ are the lower and upper frequencies in critical band z, given in Table 8.3, and $|\mathbf{C}[f]|$ is the magnitude of frequency f in the spectrum, $\mathbf{C}$.

The spreading of energy across critical bands is modeled by the *basilar membrane spreading function, SF(z)*,

$$SF(z) = 15.81 + 7.5(z + 0.474) - 17.5\sqrt{1 + (z + 0.474)^2}. \tag{8.10}$$

The apparent energy per critical band after spreading, $\mathbf{B_a}[z]$, is given by

$$\mathbf{B_a}[z] = \sum_{z'=z-z_o}^{z+z_1} \mathbf{B}[z - z']SF(z'), \tag{8.11}$$

where z_o and z_1 represent the extent of the neighborhood over which spreading is significant.

Masking depends not just on the magnitude of the apparent energy in a critical band but on whether the energy is predominantly noiselike or tonelike. To determine how noiselike or tonelike the audio frame is, we next calculate the *spectral flatness measure, SFM_{db}*, as

$$SFM_{db} = 10\log_{10}\left(\frac{\left[\prod_{z=1}^{Z_t} \mathbf{B_a}[z]\right]^{\frac{1}{Z_t}}}{\frac{1}{Z_t}\sum_{z=1}^{Z_t} \mathbf{B_a}[z]}\right) = \frac{\mu_g}{\mu_a}, \tag{8.12}$$

where μ_g and μ_a are, respectively, the geometric and arithmetic means of the power spectral density in each critical band. From the spectral flatness measure, a "coefficient of tonality," α, is derived:

$$\alpha = \min\left(\frac{SFM_{db}}{-60}, 1\right). \tag{8.13}$$

When α is close to 1, the audio window is tonelike; whereas if α is close to 0, the window is noiselike. The coefficient of tonality is then used to calculate an offset, $\mathbf{O}[z]$, given by

$$\mathbf{O}[z] = \alpha\,(14.5 + z) + (1 - \alpha)5.5. \tag{8.14}$$

The apparent energy in each critical band, $\mathbf{B_a}[z]$, is converted to decibels and the offset, $\mathbf{O}[z]$, is subtracted to obtain the masking in that band. Thus, the threshold in dB is

$$\mathbf{t_{db}}[z] = 10 \log_{10} \mathbf{B_a}[z] - \mathbf{O}[z], \tag{8.15}$$

or in Pascals,

$$\mathbf{t_{SPL}}[z] = 10^{\mathbf{t_{db}}[z]/10}. \tag{8.16}$$

To determine how much each frequency, f, within a critical band can be altered, $\mathbf{t_{SPL}}[z]$ is normalized by the number of discrete frequencies in the corresponding band. Thus, the normalized detection threshold, $\mathbf{t_n}[f]$, is given by

$$\mathbf{t_n}[f] = \frac{\mathbf{t_{SPL}}[z_f]}{N_{z_f}}, \tag{8.17}$$

where z_f is the critical band that contains frequency f, and N_{z_f} is the number of points in the critical band.

The final masking threshold, $\mathbf{t}[f]$, is obtained by comparing the normalized threshold, $\mathbf{t_n}[f]$, with the threshold of absolute hearing, $\mathbf{t_q}[f]$, as

$$\mathbf{t}[f] = \max(\mathbf{t_n}[f], \mathbf{t_q}[f]). \tag{8.18}$$

Pooling

At this point, we have the masking threshold available in each frequency. This is the minimum sound pressure level discernible. If the actual energy, $|\mathbf{C}[f]|$, present at a frequency, f, is below this threshold, $\mathbf{t}[f]$, then it is inaudible. In this case, the signal, $|\mathbf{C}[f]|$, can be altered provided that the modified value does not exceed the masking threshold. If the actual energy present exceeds the corresponding masking threshold, this model cannot indicate by how much the value can be indiscernibly changed.[3] Because the changes to one frequency are independent from changes to other frequencies, this is equivalent to pooling with an L_∞-norm.

The audio model we have described here is relatively simple. Other models are possible, including models that estimate the maximum allowable change

[3] In [160, 397], this model is used by a watermark embedder to identify all perceptibly insignificant frequencies (i.e., those frequencies for which $|\mathbf{C}[f]| \leq \mathbf{t}[f]$). The embedder then replaces these values with a watermark signal. Clearly such a procedure will not be robust to lossy compression, but may be quite practical for certain applications, particularly authentication.

in a frequency, even when the energy in that frequency exceeds the masking threshold. In these circumstances, other forms of pooling are appropriate. For example, Beerends *et al.* [33] suggests an L_1-norm or simple summation.

8.4 PERCEPTUALLY ADAPTIVE WATERMARKING

So far, this chapter has focused on perceptual models designed to measure the perceptibility of embedded watermarks. Of course, we would like to use these models to not only measure the perceptibility of marks once they are embedded but to *control* that perceptibility during the embedding process. Watermarking systems that attempt to shape the added pattern according to some perceptual model are generally referred to as *perceptually adaptive* systems. We now turn to the problem of how such systems can be created.

The simplest use of a perceptual model during embedding is for automatic adjustment of the embedding strength to obtain a particular perceptual distance. We begin this section with an example image watermark embedding method, E_PERC_GSCALE, which uses Watson's model to decide on a global embedding strength, α. The performance of this algorithm is used as a benchmark for comparison against the methods presented in the remainder of the section.

A more sophisticated use of perceptual modeling is to locally scale (or *perceptually shape*) the watermark, attenuating some areas and amplifying others, so that the watermark is better hidden by the cover Work. In Section 8.4.1, we discuss a straightforward embedding method, which multiplies each term of the watermark reference pattern by a value computed with a perceptual model. We follow this, in Section 8.4.2, with a discussion of optimal and near-optimal ways of using perceptual modeling in correlation-based watermarking systems.

INVESTIGATION

Perceptually Limited Embedding

Perceptually limited embedding describes a technique for setting the embedding strength parameter to limit the perceptual distortion introduced. We can refer to this process as a global scaling of the message pattern. This terminology will be useful when we introduce local scaling in Section 8.4.1.

System 13: E_PERC_GSCALE

The E_PERC_GSCALE algorithm is a simple modification of the E_BLIND embedding algorithm of Chapter 3 (System 1), in which the embedding strength, α, is adjusted to obtain a fixed perceptual distance, D_{target}, as measured by Watson's perceptual model. Thus, the watermarked Work will be given by

$$\mathbf{c_w} = \mathbf{c_o} + \mathbf{w_a}, \tag{8.19}$$

where $\mathbf{c_o}$ is the original unwatermarked Work and $\mathbf{w_a}$ is the added pattern. The added pattern is given by $\mathbf{w_a} = \alpha \mathbf{w}_m$, where $\mathbf{w}_m$ is a message pattern given by

$$\mathbf{w}_m = \begin{cases} \mathbf{w_r} & \text{if } m = 1 \\ -\mathbf{w_r} & \text{if } m = 0. \end{cases} \tag{8.20}$$

where $\mathbf{w_r}$ is a reference pattern.

The perceptual distance between $\mathbf{c_w}$ and $\mathbf{c_o}$, as measured by Watson's model, $D_{\text{wat}}(\mathbf{c_o}, \mathbf{c_w})$, is a linear function of α. To see this, let $\mathbf{C_w}$, $\mathbf{C_o}$, and $\mathbf{W}_m$ be the block DCT transforms of $\mathbf{c_w}$, $\mathbf{c_o}$, and $\mathbf{w}_m$, respectively. Because the block DCT is a linear transform, we have

$$\mathbf{C_w} = \mathbf{C_o} + \alpha \mathbf{W}_m. \tag{8.21}$$

According to the pooling function used in Watson's model (Equation 8.6), the perceptual distance between $\mathbf{c_w}$ and $\mathbf{c_o}$ is estimated as

$$D_{\text{wat}}(\mathbf{c_o}, \mathbf{c_w}) = \sqrt[4]{\sum_{i,j,k} \left(\frac{\mathbf{C_w}[i,j,k] - \mathbf{C_o}[i,j,k]}{\mathbf{s}[i,j,k]} \right)^4}, \tag{8.22}$$

where $\mathbf{s}$ is an array of slacks based on estimated sensitivity functions of the eye and the masking properties of $\mathbf{c_o}$. Substituting $\alpha \mathbf{W_m}[i,j,k]$ for $\mathbf{C_w}[i,j,k] - \mathbf{C_o}[i,j,k]$ gives

$$D_{\text{wat}}(\mathbf{C_o}, \mathbf{C_w}) = \sqrt[4]{\sum_{i,j,k} \left(\frac{\alpha \mathbf{W}_m[i,j,k]}{\mathbf{s}[i,j,k]} \right)^4} \tag{8.23}$$

$$= \alpha \sqrt[4]{\sum_{i,j,k} \left(\frac{\mathbf{W}_m[i,j,k]}{\mathbf{s}[i,j,k]} \right)^4} \tag{8.24}$$

$$= \alpha D_{\text{wat}}(\mathbf{c_o}, \mathbf{c_o} + \mathbf{w}_m). \tag{8.25}$$

Thus, to set $D_{\text{wat}}(\mathbf{c_o}, \mathbf{c_w})$ equal to a desired perceptual distance, D_{target}, we obtain α as

$$\alpha = \frac{D_{\text{target}}}{D_{\text{wat}}(\mathbf{c_o}, \mathbf{c_o} + \mathbf{w_r})}. \tag{8.26}$$

In practice, round-off and clipping can cause watermarks embedded with this α to have different numbers of JNDs. To mitigate this problem, we perform an exhaustive search of values between 0.2α and 1.1α, looking for the one that, with round-off and clipping, yields the closest value to the desired number of JNDs.

Experiments

Using this embedding algorithm, we embedded messages of 0 and 1 in 2,000 images with a white-noise reference pattern and $D_{target} = 4$. The distribution of perceptual distances between the watermarked and unwatermarked versions of these images, as measured by Watson's model, is shown in Figure 8.11. This shows that the algorithm achieved a tight distribution around a distance of 4. The deviations from this distance result from round-off and clipping errors.

The distribution of linear correlations between the watermarked images and the reference pattern is shown in Figure 8.12. Note that at this fidelity constraint the algorithm achieves very poor separation between the detection values for watermarked and unwatermarked images. These values are considerably worse than values achieved (with the same limit on perceptual distance) by embedding algorithms developed in the following sections.

The E_PERC_GSCALE algorithm and results are provided here for baseline comparison.

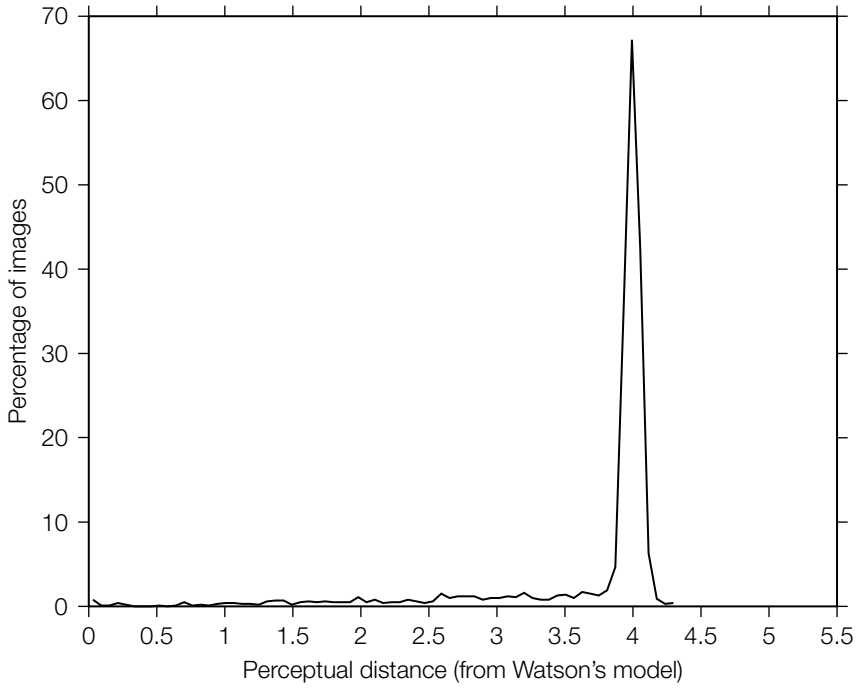


FIGURE 8.11

Histogram of perceptual distances obtained using the E_PERC_GSCALE embedding algorithm, with a target perceptual distance of 4. Results of tests on 2,000 images.

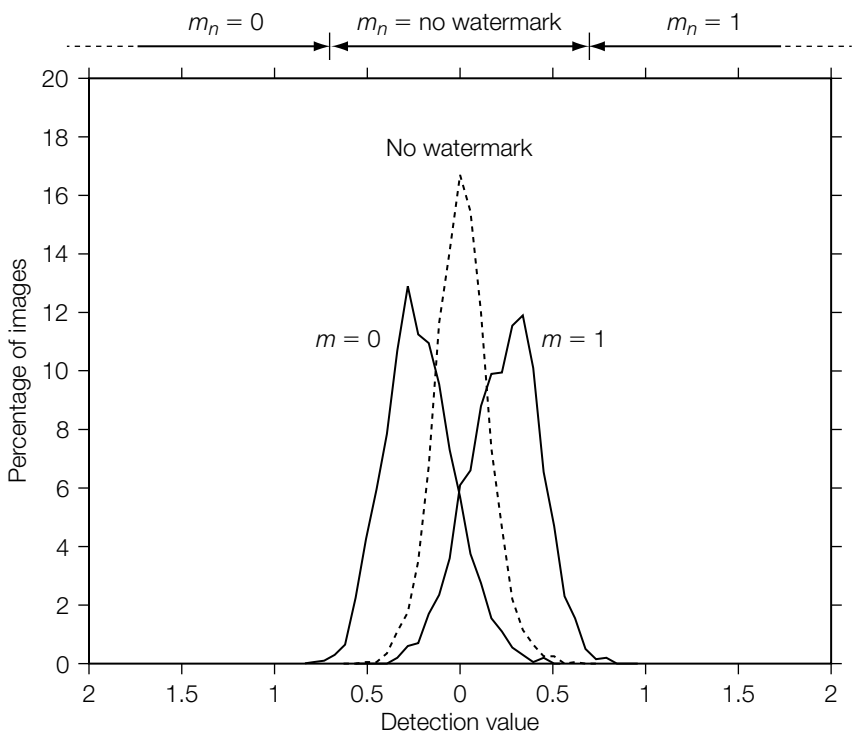


FIGURE 8.12

Histograms of detection values obtained using the E_PERC_GSCALE embedder with a target perceptual distance of 4. Results of tests on 2,000 images.

8.4.1 Perceptual Shaping

Intuitively, we should be able to embed a stronger watermark if we amplify the mark in the areas where it is well hidden (e.g., the mountains in Figure 8.8) and attenuate it in areas where it is plainly perceptible (e.g., the sky in Figure 8.8). We refer to such a strategy as *perceptual shaping*. Some examples of the application of this idea to image watermarking are given in [29, 108, 335, 395].

To apply perceptual shaping in a watermark embedder, we need a model that assigns a perceptual slack, $\mathbf{s}[i]$, to each term of the cover Work, expressed in some domain. In the case of Watson's model, a slack is assigned to each term of an image in the block DCT domain. Other perceptual models for images assign slacks to pixels in the spatial domain [429], frequencies in the Fourier domain, or terms in various wavelet domains [437]. Similarly, perceptual models for audio exist in the time, Fourier, Bark, and wavelet domains [318]. After obtaining the slacks with the model, we transform the message pattern into the

domain of the perceptual model, and scale each term by its slack. Transforming back into the temporal or spatial domain yields a *shaped pattern* that can be well hidden in the cover Work. A simple version of this embedding process, designed for a watermarking system that uses media space as its marking space, is illustrated in Figure 8.13.

In several implementations of the perceptual shaping idea, the shaping applied in the embedder is inverted at the detector [337]. This is shown in Figure 8.14 for an informed detector. Given the original Work, the detector's perceptual model computes the exact slacks by which the message pattern was scaled during embedding. A watermark pattern, extracted from the watermarked Work by subtracting the original Work, is then transformed into the domain of the perceptual model, divided by the slacks, and transformed back into media space. If the watermarked Work was not distorted or modified in any way between embedding and detection, the resulting pattern is exactly the reference pattern used by the embedder. Detection can then proceed as before, with a comparison between the unscaled, extracted watermark and the reference watermark. Alternatively, the slacks can be used to scale the

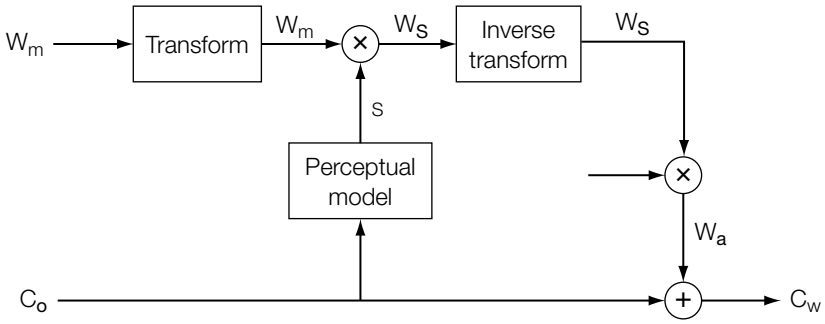


FIGURE 8.13

Basic design of an embedder that uses perceptual shaping. (In this embedder design, marking space is the same as media space.)

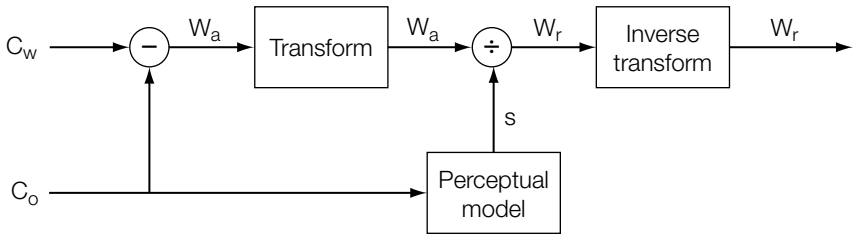


FIGURE 8.14

Basic design of an informed detector that inverts perceptual shaping.

message pattern in the detector, and thus provide a more appropriate signal for comparison with the extracted pattern.

Although inversion of perceptual shaping works best with informed detection, it can be approximated in a blind detector as well [90, 337]. One approach is to apply the perceptual model to the received (possibly watermarked and noisy) Work. Assuming that the application of the watermark to the cover Work is perceptually transparent, the watermarked Work and the original Work should have similar masking characteristics, and the slacks derived from the two should be similar. Thus, the informed detection system of Figure 8.14 can be applied in a blind detector by substituting the received Work for the cover Work.

Inversion of perceptual shaping in the detector is not always necessary. Instead, the detector may disregard the embedder's use of perceptual shaping. A system designed in this way essentially views perceptual shaping as a distortion of the watermark. If watermark detection is robust to the noise introduced by this shaping, a detector that does not invert the shaping will still detect the watermark.

Ordinarily, a distortion of the embedded watermark, such as that caused by perceptual shaping, would reduce its detectability. This suggests that perceptual shaping in the embedder, without the corresponding inversion in the detector, should lead to lower detection values. However, perceptual shaping allows the watermark to be embedded with greater strength, and the net result is a mark that gives a higher detection value. For example, consider a linear correlation system with a message pattern, $\mathbf{w}_m$, which, after shaping, results in some *shaped pattern*,[4] $\mathbf{w}_s$. This shaped pattern can be viewed as a distorted, scaled version of the message pattern:

$$\mathbf{w}_s = \gamma \mathbf{w}_m + \mathbf{g}, \tag{8.27}$$

where γ is a scalar and $\mathbf{g}$ is a vector orthogonal to $\mathbf{w}_m$. The vector $\mathbf{g}$ can be thought of as a "masking" pattern, which distorts the reference mark to make it less perceptible. Suppose $\gamma = 1/2$. Adding $\mathbf{w}_s$ to a cover Work would lead to half the linear correlation we would get by adding $\mathbf{w}_m$ to the Work. However, because of the perceptual shaping we might be able to more than double the energy of $\mathbf{w}_s$ before the fidelity becomes comparable with that of adding $\mathbf{w}_m$. That is, if

$$\mathbf{c}_{ws} = \mathbf{c}_o + \alpha_s \mathbf{w}_s \tag{8.28}$$

and

$$\mathbf{c}_{wm} = \mathbf{c}_o + \alpha_m \mathbf{w}_m, \tag{8.29}$$

then the perceptual quality of $\mathbf{c}_{ws}$ might be as good as or better than that of $\mathbf{c}_{wm}$, even for values of α_s greater than $2\alpha_m$ (because $\mathbf{w}_s$ is better hidden by $\mathbf{c}_o$ than is $\mathbf{w}_m$). The linear correlation between $\mathbf{c}_{ws}$ and $\mathbf{w}_m$ is

[4] The shaped pattern is not necessarily the added pattern, in that it might be scaled before being added.

$$z_{\text{lc}}(\mathbf{c_{ws}}, \mathbf{w}_m) = \frac{1}{N}(\mathbf{c_o} + \alpha_s \mathbf{w_s}) \cdot \mathbf{w}_m$$

$$= \frac{1}{N}(\mathbf{c_o} + \alpha_s \gamma \mathbf{w}_m + \alpha_s \mathbf{g}) \cdot \mathbf{w}_m$$

$$= \frac{1}{N}\mathbf{c_o} \cdot \mathbf{w_r} + \alpha_s \gamma \mathbf{w}_m \cdot \mathbf{w}_m. \tag{8.30}$$

If $\gamma(\alpha_s / \alpha_m) > 1$, this is higher than the correlation between $\mathbf{c_{wm}}$ and $\mathbf{w}_m$; namely,

$$z_{\text{lc}}(\mathbf{c_{wm}}, \mathbf{w}_m) = \frac{1}{N}\mathbf{c_o} \cdot \mathbf{w}_m + \mathbf{w}_m \cdot \mathbf{w}_m. \tag{8.31}$$

Figure 8.15 shows a geometric interpretation of this point. The figure illustrates a two-dimensional (2D) slice through media space that contains both the message pattern, $\mathbf{w}_m$, and the unwatermarked Work, $\mathbf{c_o}$. The *x*-axis is aligned with the reference pattern. The ellipse represents a region of acceptable fidelity, as measured by some perceptual model. We see that only $1 \times \mathbf{w}_m$ can be

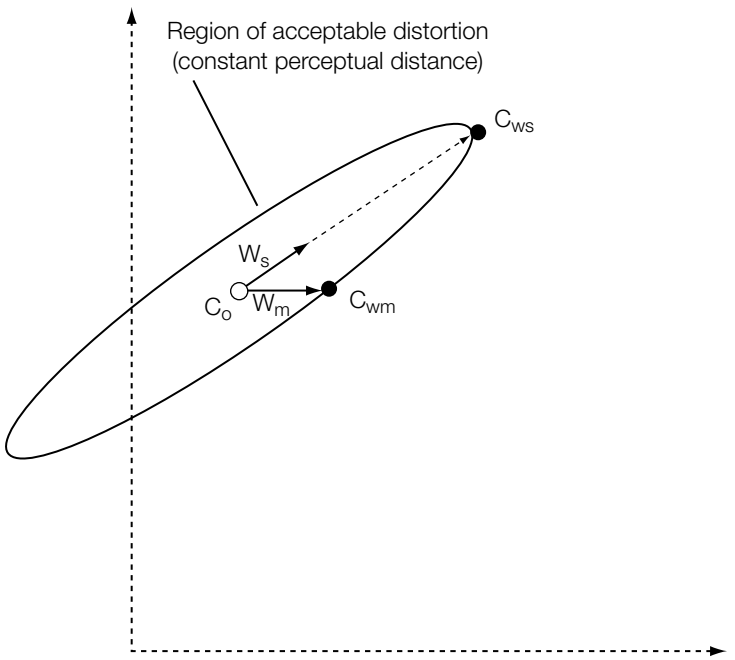


FIGURE 8.15

Geometric explanation that perceptual shaping can result in stronger detection. This is a 2D slice of media space, with the *x*-axis aligned with the message pattern, $\mathbf{w}_m$. Even though the perceptually scaled pattern, $\mathbf{w_s}$, is not perfectly aligned with $\mathbf{w}_m$, it may be scaled by a much larger value before reaching the same level of perceptual distortion. This results in a point further along the *x*-axis and, hence, with a higher linear correlation.

added to $\mathbf{c_o}$ before reaching the limit of acceptable distortion, leading to one possible watermarked Work, $\mathbf{c_{wm}}$. Applying a perceptual shaping algorithm to $\mathbf{w}_m$, with reference to $\mathbf{c_o}$, might result in the vector $\mathbf{w_s}$. Although the correlation between $\mathbf{w_s}$ and $\mathbf{w}_m$ is less than one, we can scale $\mathbf{w_s}$ by a large amount before reaching the perceptual limit. The resulting watermarked Work, $\mathbf{c_{ws}}$, has much higher correlation with $\mathbf{w}_m$ than does $\mathbf{c_{wm}}$.

INVESTIGATION

Perceptually Shaped Embedding

We now present a 1-bit image watermark embedding algorithm, E_PERC_SHAPE, which employs Watson's perceptual model for perceptual shaping. This algorithm is similar to that proposed by Podilchuk and Zeng [335]. The algorithm is designed to embed with a constant perceptual distance, as measured by that model. This is accomplished by choosing the embedding strength, α, in the same manner as in the E_PERC_GSCALE algorithm (Equation 8.26), with the exception of replacing the message mark, $\mathbf{w}_m$, with a perceptually shaped mark, $\mathbf{w_s}$.

System 14: E_PERC_SHAPE

The E_PERC_SHAPE algorithm begins, like most other examples we have presented, by using the message, m, to determine the sign of the pattern it will embed, $\mathbf{w}_m$. It differs from the other embedding algorithms in that it now uses the Watson model to perceptually shape $\mathbf{w}_m$. First, it computes the block DCT, $\mathbf{C_o}$, of the cover Work and applies Equations 8.3 and 8.5 to obtain an array of slacks, $\mathbf{s}$. Each slack, $\mathbf{s}[i, j, k]$, estimates the amount by which the i, jth term of block k may be changed before leading to one JND. Next, the embedder takes the block DCT, $\mathbf{w}_m$, of the message pattern and multiplies each term by the corresponding slack:

$$\mathbf{w_s}[i, j, k] = \mathbf{w}_m[i, j, k]\mathbf{s}[i, j, k]. \tag{8.32}$$

Applying the inverse block DCT to $\mathbf{w_s}$ yields the perceptually shaped version of the reference pattern, $\mathbf{w_s}$. Finally, to obtain the watermarked Work, the shaped pattern is added to the original Work after being multiplied by a scaling factor, α, as

$$\mathbf{c_w} = \mathbf{c_o} + \alpha\mathbf{w_s}. \tag{8.33}$$

The scaling factor is given by a local search around

$$\alpha = \frac{D_{\text{target}}}{D_{\text{wat}}(\mathbf{c_o}, \mathbf{c_o} + \mathbf{w_a})}, \tag{8.34}$$

where D_{target} is a target perceptual distance entered as a parameter.

Experiment

Figure 8.16 shows the result of using the E_PERC_SHAPE algorithm to embed a watermark in an image. The target perceptual distance here was chosen to obtain roughly the same linear correlation with the reference mark as obtained from Figure 8.8(b). (Figure 8.16 gives a detection value of 3.83, and Figure 8.8(b) gives a detection value of 3.91.) Figure 8.17 shows the difference between the original image and the image watermarked by E_PERC_SHAPE. In these figures we see that the E_PERC_SHAPE algorithm suppresses the noise that is visible in the sky in Figure 8.8(b). To compensate for the loss of detection strength in the sky, the E_PERC_SCALE algorithm has amplified the watermark in the mountainous area, where it is not as visible. Of course, at this level of embedding strength, the watermark is still visible, primarily in the form of blocking artifacts around the edges of the mountains. These artifacts result from the block-based design of the perceptual model, which means that it underestimates the perceptibility of edges along block boundaries.

In actual use, we would not embed with such large perceptual distances. Figure 8.18 shows the distributions of detection values obtained by embedding watermarks in 2,000 images with the E_PERC_SHAPE algorithm, using a target

FIGURE 8.16

Image with watermark embedded by E_PERC_SHAPE algorithm. This yields roughly the same detection value as the image in Figure 8.8(b), which was created by the E_BLIND algorithm.

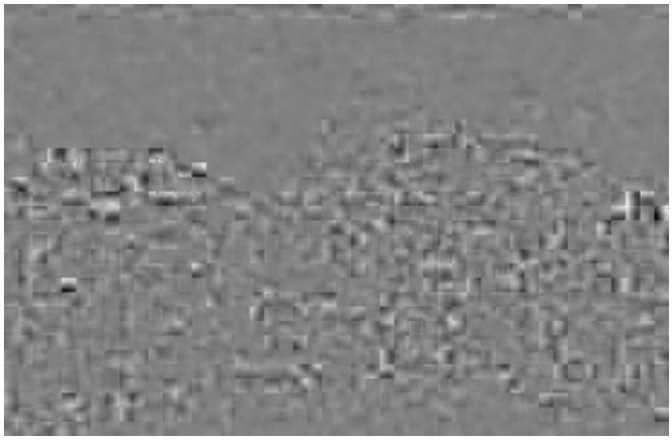


FIGURE 8.17

Perceptually shaped watermark that is added by the E_PERC_SHAPE algorithm to create Figure 8.16. Note that there is less energy in the area of the sky than in the area of the mountains.

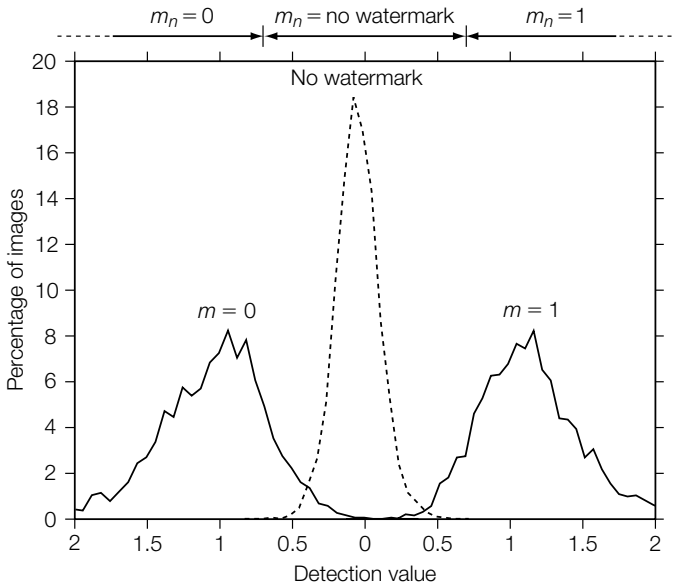


FIGURE 8.18

Detection results obtained after embedding with the E_PERC_SHAPE algorithm and a perceptual limit of four JNDs, as measured by Watson's model. Results from tests on 2,000 images.

perceptual distance of $D_{\text{target}} = 4$. These results should be compared against those in Figure 8.12. This shows that for a fixed perceptual distance, perceptual shaping greatly increases the detection value.

8.4.2 Optimal Use of Perceptual Models

The perceptual shaping strategy used by the E_PERC_SHAPE algorithm is founded on the intuitive notion that the watermark should be amplified in some areas and attenuated in others. Of course, there are other ways we might use a perceptual model to shape the watermark, rather than the linear scaling described so far. For example, we might choose an amplitude for the reference pattern and then clip each term by the corresponding slack assigned by the model. This would ensure that no term is changed by more than one (estimated) JND. Alternatively, we might apply some nonlinear scaling, multiplying by slacks raised to a constant power. But what is the optimal use of a given perceptual model in embedding watermarks?

Depending on the application, we can specify two alternative optimal behaviors for the embedder. On the one hand, we might want the embedder to maximize the robustness of an embedded mark, while maintaining a constant perceptual distance. On the other hand, we might want to minimize perceptual distance while maintaining a constant robustness. In the case of linear correlation systems, we can replace the word *robustness* in these two specifications with *detection value,* in that increasing linear correlation generally increases robustness. In the case of normalized correlation, we should use a separate robustness measure, such as the R^2 value described in Chapter 5.

For either of these two notions of optimality, in some cases it is possible to analytically compute the optimal pattern to be added. We begin by considering such a case; namely, a linear correlation system with a perceptual model that uses a scaled L_p-norm for pooling.[5]

Optimal Perceptual Shaping in a Linear Correlation System

Consider a watermarking system using a blind, linear correlation–based detector. Assume that this detector operates directly in media space. Therefore, the detection value it computes is just

$$z_{\text{lc}}(\mathbf{c}, \mathbf{w}_m) = \frac{1}{N}\mathbf{c} \cdot \mathbf{w}_m, \tag{8.35}$$

[5] Recently, Altun *et al.* [19] have proposed a set theoretic framework for watermarking in which requirements such as fidelity and robustness are described as constraint sets. They show that in some cases these constraints can be formulated as convex sets and feasible solutions found using the method of projection on to convex sets.

representing the likelihood that work $\mathbf{c}$ contains mark $\mathbf{w}_m$. The D_LC detection algorithm of Chapter 3 is an example of such a detector. If we are given a perceptual model that estimates perceptual distance between two Works, $D(\mathbf{c_o}, \mathbf{c_w})$, we can try to build an optimal embedder for this system, where optimality is specified in either of the ways discussed previously. That is, the embedder might either try to maximize the correlation between the watermarked Work and the reference pattern, while keeping the perceptual distance constant, or try to minimize perceptual distance while keeping the correlation constant. We examine the former case first.

Fixed Fidelity

We want to maximize $z_{\text{lc}}(\mathbf{c_w}, \mathbf{w}_m)$ while keeping $D(\mathbf{c_o}, \mathbf{c_w})$ constant. Equivalently, we can maximize $z_{\text{lc}}(\mathbf{w_a}, \mathbf{w}_m)$ while keeping $D(\mathbf{c_o}, \mathbf{c_o} + \mathbf{w_a})$ constant, where $\mathbf{w_a}$ is the added pattern.

Many perceptual distance measures can be reduced to the form

$$D(\mathbf{c_o}, \mathbf{c_w}) = \sqrt[P]{\sum_i \left(\frac{\mathbf{C_w}[i] - \mathbf{C_o}[i]}{\mathbf{s}[i]} \right)^P}, \tag{8.36}$$

where $\mathbf{C_w}$ and $\mathbf{C_o}$ are $\mathbf{c_w}$ and $\mathbf{c_o}$ after application of some energy-preserving linear transform, such as the DCT or Fourier transform, and where $\mathbf{s}$ is an array of slack values based on the human sensitivity functions and the masking properties of $\mathbf{c_o}$. Assuming such a perceptual model, we can rephrase the constraint on our optimization problem as

$$D(\mathbf{c_o}, \mathbf{c_o} + \mathbf{w_a})^P = \sum_i \left(\frac{\mathbf{W_a}[i]}{\mathbf{s}[i]} \right)^P = D_{\text{target}}^P, \tag{8.37}$$

where $\mathbf{W_a}$ is the transform of the added pattern and D_{target}^P is a constant. Because the correlation between two vectors is the same as the correlation between transformed versions of them (assuming an energy-preserving linear transform), we can rephrase the objective of our optimization problem as maximizing $z = z_{\text{lc}}(\mathbf{W_a}, \mathbf{W}_m)$.

The solution to this optimization problem can be found with the technique of Lagrange multipliers by solving the equation

$$F' - \lambda G' = 0, \tag{8.38}$$

where

$$F' = \frac{\partial z_{\text{lc}}}{\partial \mathbf{W_a}[i]} = \mathbf{W}_m[i] \tag{8.39}$$

$$G' = \frac{\partial D_{\text{target}}^P}{\partial \mathbf{W_a}[i]} = P \frac{\mathbf{W_a}[i]^{P-1}}{\mathbf{s}[i]^P}. \tag{8.40}$$

Solving Equation 8.38 for $\mathbf{W_a}[i]$ yields

$$\mathbf{W_a}[i] = \left(\frac{\mathbf{W}_m[i]\mathbf{s}[i]^P}{\lambda P} \right)^{\frac{1}{P-1}} \tag{8.41}$$

$$= \alpha \left(\mathbf{W}_m[i]s[i]^P \right)^{\frac{1}{P-1}}, \tag{8.42}$$

where

$$\alpha = \left(\frac{1}{\lambda P} \right)^{\frac{1}{P-1}} \tag{8.43}$$

and a specific choice of λ will depend on the particular choice of D_{target} in Equation 8.37. We can compute the optimal $\mathbf{w_a}$ by first computing a shaped pattern, $\mathbf{W_s}$, as

$$\mathbf{W_s}[i] = \left(\mathbf{W_m}[i]\mathbf{s}[i]^p \right)^{\frac{1}{p-1}} \tag{8.44}$$

and applying the inverse transform to obtain $\mathbf{w_s}$. Next, compute the correct value of α as

$$\alpha = \frac{D_{\text{target}}}{D\left(\mathbf{c_o}, \mathbf{c_o} + \mathbf{w_s}\right)} \tag{8.45}$$

and let

$$\mathbf{w_a} = \alpha \mathbf{w_s}. \tag{8.46}$$

Fixed Linear Correlation

Now suppose that instead of maximizing linear correlation for a given perceptual distance we want to minimize perceptual distance for a given linear correlation. Therefore, we want to minimize Equation 8.37 while satisfying the constraint

$$z_{\text{lc}}(\mathbf{c_w}, \mathbf{w}_m) = z_{\text{target}}, \tag{8.47}$$

where z_{target} is a constant. This constraint can be rewritten as

$$\begin{aligned} z_{\text{lc}}(\mathbf{c_o} + \mathbf{w_a}, \mathbf{w}_m) &= z_{\text{lc}}(\mathbf{c_o}, \mathbf{w}_m) + z_{\text{lc}}(\mathbf{w_a}, \mathbf{w}_m) \\ &= z_{\text{lc}}(\mathbf{c_o}, \mathbf{w}_m) + z_{\text{lc}}(\mathbf{W_a}, \mathbf{W}_m) = z_{\text{target}}. \end{aligned} \tag{8.48}$$

Because $z_{\text{lc}}(\mathbf{c_o}, \mathbf{w}_m)$ is constant, the constraint on $z_{\text{lc}}(\mathbf{W_a}, \mathbf{W}_m)$ is

$$z_{\text{lc}}(\mathbf{W_a}, \mathbf{W}_m) = z'_{\text{target}}, \tag{8.49}$$

where $z'_{\text{target}} = z_{\text{target}} - z_{\text{lc}}(\mathbf{c_o}, \mathbf{w}_m)$. This optimization problem can be solved with Lagrange multipliers in the same manner as the previous problem,

and the solution has essentially the same form. Specifically, $\mathbf{w_a}$ is given by Equations 8.44 and 8.46, but with a different value of α. In this case, after computing $\mathbf{w_s}$, instead of computing α with Equation 8.45 we compute it as

$$\begin{aligned}\alpha &= \frac{z'_{\text{target}}}{z_{\text{lc}}(\mathbf{w_s}, \mathbf{w}_m)} \\ &= \frac{z_{\text{target}} - z_{\text{lc}}(\mathbf{c_o}, \mathbf{w}_m)}{z_{\text{lc}}(\mathbf{w_s}, \mathbf{w}_m)}.\end{aligned} \tag{8.50}$$

INVESTIGATION

Optimally Scaled Embedding

We now describe a watermark embedding algorithm that maximizes the linear correlation for a fixed fidelity, where fidelity is measured by the Watson model.

System 15: E_PERC_OPT

The E_PERC_OPT embedding algorithm is essentially the same as the E_PERC_SHAPE algorithm, except that instead of linearly scaling the block DCT terms of the reference pattern by their respective slacks, we use the optimal formula described previously. In the Watson perceptual model, pooling is performed by an L_4-norm; thus,

$$D_{\text{wat}}(\mathbf{c_o}, \mathbf{c_o} + \mathbf{w_a}) = \sqrt[4]{\sum_{i,\ j,\ k} \left(\frac{\mathbf{W_a}[i,\ j,\ k]}{\mathbf{s}[i,\ j,\ k]} \right)^4}, \tag{8.51}$$

where $\mathbf{W_a}[i,\ j,\ k]$ is the $i,\ j$th term of the kth block in the block DCT of $\mathbf{w_a}$, and $\mathbf{s}[i,\ j,\ k]$ is the corresponding slack. This means we can use Equation 8.44 with $p = 4$ to find the optimal perceptual shaping of the added mark. This gives

$$\mathbf{W_s}[i,\ j,\ k] = (\mathbf{W}_m[i,\ j,\ k]\mathbf{s}[i,\ j,\ k]^4)^{\frac{1}{3}}. \tag{8.52}$$

Embedding is then completed by computing the inverse block DCT of $\mathbf{W_s}$ to obtain $\mathbf{w_s}$, determining α according to Equation 8.45, and letting $\mathbf{c_w} = \mathbf{c_o} + \alpha\mathbf{w_s}$.

Experiment

Figure 8.19 plots the distribution of detection values obtained using the E_PERC_OPT algorithm, with $D_{\text{target}} = 4$. Comparing these results with those in Figure 8.18 (included here with dotted lines) shows that the E_PERC_OPT algorithm generally leads to stronger detection values than does E_PERC_SHAPE for the same perceptual distance. Although the improvement in performance shown by Figure 8.19 may not appear large, it can have a noticeable effect on

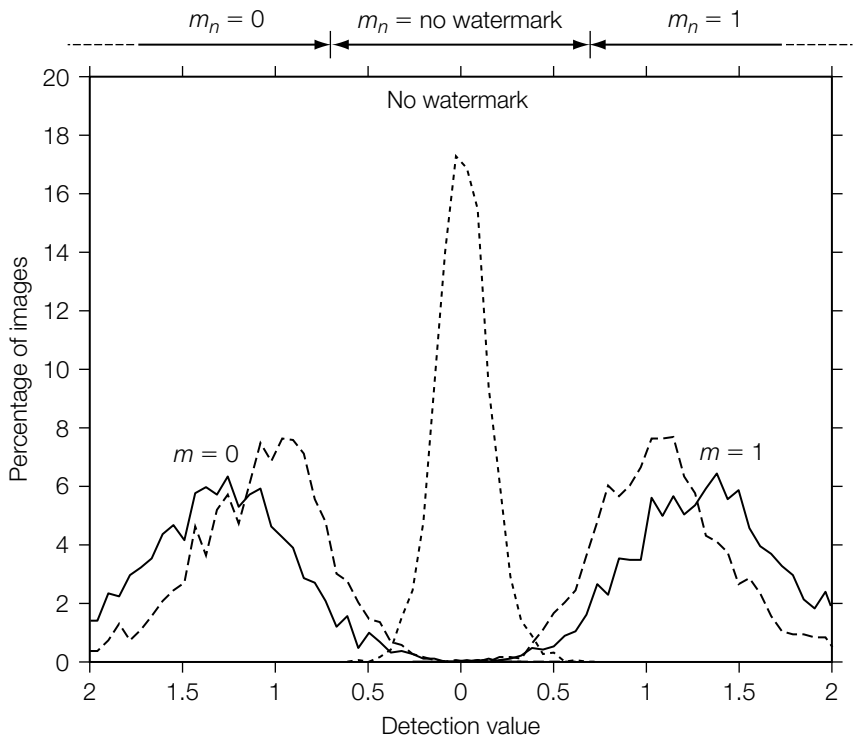


FIGURE 8.19

Detection results obtained after embedding with the E_PERC_OPT algorithm and a perceptual limit of four JNDs (as measured by Watson's model). The results for the E_PERC_SHAPE algorithm are included for comparison (dotted lines). Results are from tests on 2,000 images.

the overall effectiveness of the embedding algorithm. For example, with a threshold of 0.7, the E_PERC_SHAPE algorithm succeeds in embedding watermarks in about 89% of the images. This goes up to 95% with the E_PERC_OPT algorithm.

Optimal Perceptual Shaping in a Normalized Correlation System

If our watermarking system uses normalized correlation as its detection statistic, the optimal use of a perceptual model is more complex than for linear correlation systems. As pointed out in Chapter 5, and illustrated by E_BLK_FIXED_CC/D_BLK_CC (System 7), we cannot assume that high detection values correspond to high robustness. Instead, we must use a different measure

of robustness, such as the R^2 value used in E_BLK_FIXED_R (System 8). However, even in the simple case of the E_BLK_FIXED_R algorithm, which tries to minimize the Euclidian distance between the watermarked and unwatermarked versions of the Work, the R^2 measure leads to a problem that is complex to solve analytically. In that algorithm we resort to a brute-force search to find a near-optimal solution. When we replace Euclidian distance with a more sophisticated perceptual distance function, the optimization problem is almost certain to become more difficult.

In a normalized correlation system with a sophisticated perceptual model, one possible way of finding an optimal (or near-optimal) shaped pattern is to use a general search technique, such as simulated annealing, gradient descent, or the simplex method [336]. If the system works in media space, or in any marking space with very high dimensionality, the number of independent variables may make direct search prohibitive. For example, to watermark a 720×480 pixel image we would need to solve a problem with 345,600 variables. This is more than most general search techniques can handle.

An alternative suboptimal solution is to embed the watermark as though we are using linear correlation. This relies on the fact that, generally, increasing linear correlation increases R^2. To see why, note that in media space R^2 is defined as

$$R^2 = \left(\frac{\mathbf{c_w} \cdot \mathbf{w}_m}{\tau_{\text{nc}} \|\mathbf{w}_m\|} \right)^2 - \mathbf{c_w} \cdot \mathbf{c_w}, \tag{8.53}$$

where $\mathbf{c_w}$ is the proposed watermarked Work, $\mathbf{w}_m$ is the watermark message pattern, and τ_{nc} is the threshold we expect the detector to use. If perceptual shaping increases the linear correlation between $\mathbf{c_w}$ and $\mathbf{w}_m$ without making too large a difference in $\mathbf{c_w} \cdot \mathbf{c_w}$, it increases R^2. Thus, an algorithm (such as E_PERC_OPT) that maximizes $\mathbf{c_w} \cdot \mathbf{w}_m$ should generally lead to large values of R^2.

INVESTIGATION

The Effect of Perceptually Shaped Embedding on Detection Using Normalized Correlation

We now present a brief investigation to test the claim that the E_PERC_OPT embedder generally increases robustness for a correlation coefficient detector, D_BLK_CC, even though the embedder is optimized for linear correlation.

Experiment

To test this claim, we used E_PERC_GSCALE and E_PERC_OPT to embed a watermark in 2,000 images with a tiled reference pattern, and a constant

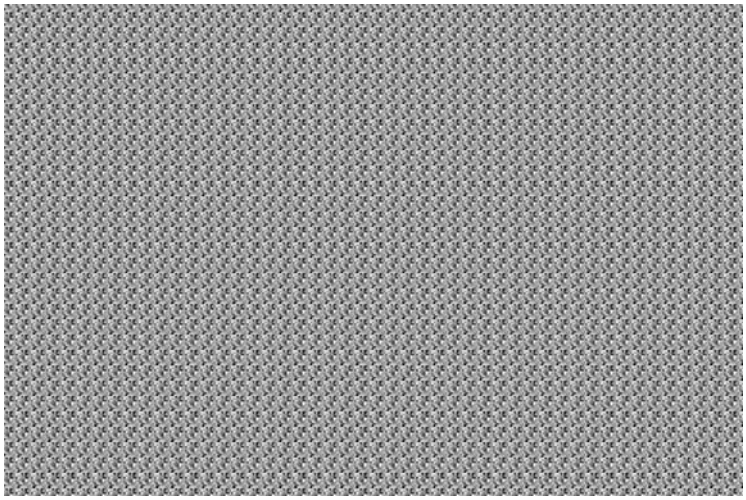


FIGURE 8.20

Tiled watermark pattern used to test effectiveness of E_PERC_OPT as an embedder when watermarks are detected by the D_BLK_CC detection algorithm.

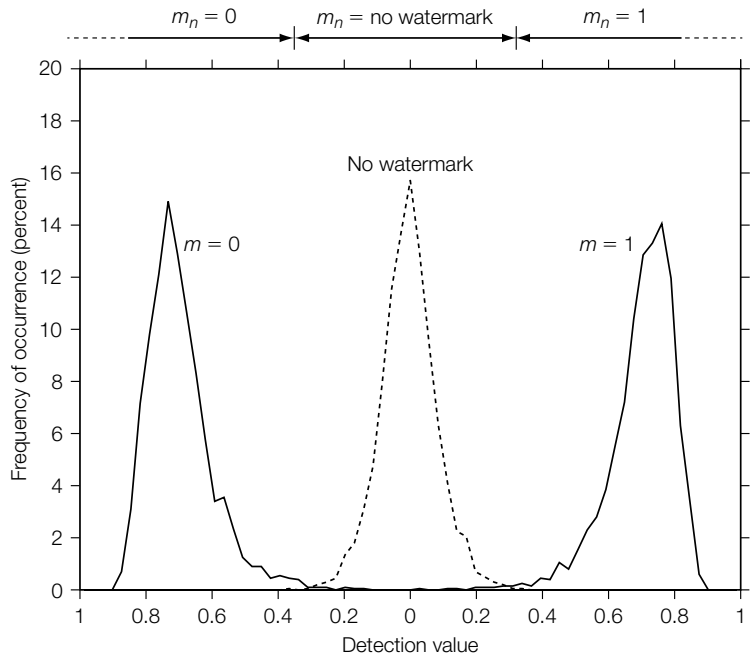


FIGURE 8.21

Detection results obtained after embedding a tiled pattern with the E_PERC_OPT algorithm and detecting with the D_BLK_CC detector. Results from tests on 2,000 images.

perceptual distance of 4. Each 8 × 8 block of this reference pattern is identical to all others, as shown in Figure 8.20. As pointed out in Chapter 3, such a reference pattern should be detectable by the D_BLK_CC detection algorithm, which extracts watermarks by summing 8 × 8 blocks, and then uses the correlation coefficient to compare them against a reference mark.

Figure 8.21 shows the distributions of detection values computed in D_BLK_CC, indicating that the embedded watermark is, in fact, detected by that algorithm. Figure 8.22 shows that the watermarks embedded by E_PERC_OPT tend to have higher robustness, as measured with R^2, than do those embedded by E_PERC_GSCALE. This means that the perceptual shaping performed by E_PERC_OPT generally increases R^2, even though E_PERC_OPT is optimized for linear correlation.

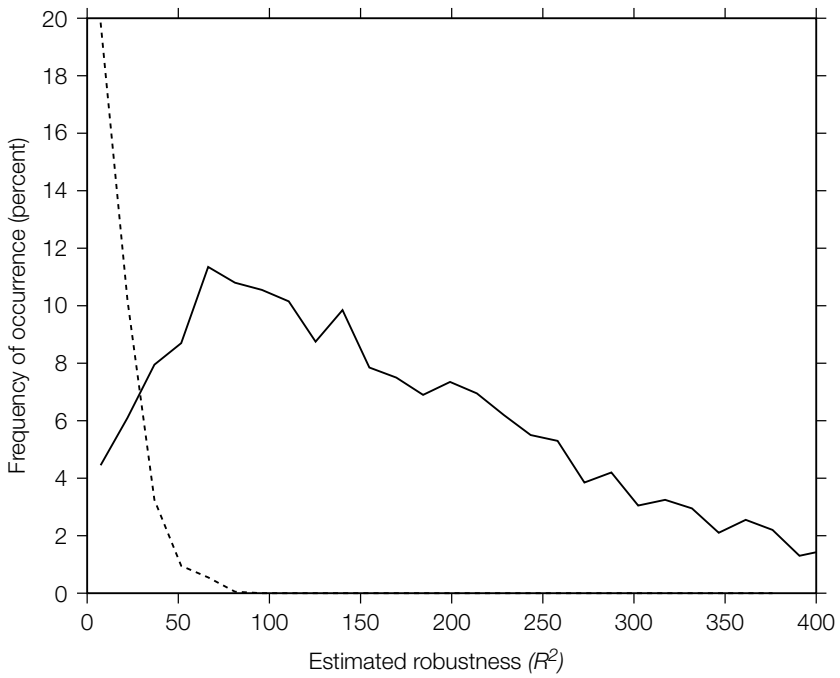


FIGURE 8.22

Comparison of R^2 robustness values obtained by embedding with E_PERC_GSCALE (dotted line) and E_PERC_OPT (solid line). Images in which watermarks were not successfully embedded are not included in this graph. Results from tests on 2,000 images.

8.5 SUMMARY

Perceptual modeling is an important aspect of a watermarking system. The main points of this chapter were the following.

- Perceptibility can be based on either fidelity or quality:
  - Fidelity measures the relative similarity between two signals.
  - Quality is an absolute measure of appeal.
- The just noticeable difference or JND is a psychophysical measure of the level of distortion that can be perceived in 50% of experimental trials.
- Experimental trials are often performed using a two alternative forced choice (2AFC).
- Human trials are expensive and automated evaluations are often preferred based on perceptual models of the human visual and auditory systems.
- Mean square error (MSE) is a poor perceptual model.
- More sophisticated perceptual models generally account for three basic phenomena: sensitivity, masking, and pooling:
  - Sensitivity measures an observer's response to isolated stimuli.
  - Masking measures an observer's response to one stimulus when a second "masking" stimulus is also present.
  - Pooling combines the perceptibilities of separate distortions to give a single estimate of the overall change in the Work.
- Two examples of perceptual modeling were described: Watson's DCT-based visual model and the MPEG-1 Layer 1 audio model.
- Perceptual modeling can be used in a variety of ways for perceptually adaptive watermarking:
  - The perceptual distance can be used to determine a global scaling factor.
  - Perceptual shaping allows for local amplification and attenuation depending on the local properties of the cover Work.
  - In the case of a linear correlation detector, a perceptual model can be used to find the optimal mark to embed.
  - For normalized correlation, the optimization is more difficult. However, unlike System 7, the `E_PERC_OPT` embedder also benefits normalized correlation.

This page intentionally left blank

CHAPTER 9

Robust Watermarking

Many applications require watermarks to be detected in Works that may have been altered after embedding. Watermarks designed to survive legitimate and everyday usage of content are referred to as *robust* watermarks. In this chapter, we present several general methods for making watermarks robust and discuss specific methods for handling some of the most common types of processing.

We draw a distinction between robust watermarks and *secure* watermarks.[1] Whereas robust watermarks are designed to survive normal processing, secure watermarks are designed to resist any attempt by an adversary to thwart their intended purpose. Because in most applications a watermark cannot perform its function if it is rendered undetectable, robustness is a necessary property if a watermark is to be secure. In other words, if a watermark can be removed by application of normal processes, it cannot be considered secure. However, robustness is not sufficient for security, because secure watermarks must be also capable of surviving novel processes that are specifically designed to remove them. Thus, the designer of a secure watermark must consider the range of *all possible* attacks. The designer of a robust watermark can limit his or her attention to the range of *probable* processing. Secure watermarks are discussed in Chapter 10.

In designing a robust watermark it is important to identify the specific processes that are likely to occur between embedding and detection. Examples of processes a watermark might need to survive include lossy compression, digital-to-analog-to-digital conversion, analog recording (such as VHS or audio tape), printing and scanning, audio playback and re-recording, noise reduction, format conversion, and so on. However, robustness to a given process often comes at some expense in computational cost, data payload, fidelity, or even robustness to some other process. It is therefore wise to ignore those processes that are

[1] Not all authors make this distinction. Instead, they use the term *robustness* to refer to the ability to survive *all* forms of distortion, hostile or otherwise. We believe, however, that resistance to hostile attacks is technically very different from resistance to normal processing, and we therefore use *robustness* to refer exclusively to the latter.

unlikely in a given application. For example, a video watermark designed for monitoring television advertisements will need to survive the various processes involved in broadcasting—digital-to-analog conversion, lossy compression, and so on—but need not survive other processes, such as rotation or halftoning.

Section 9.1 of this chapter describes several general approaches to making watermarks robust. Some of these aim to make the watermark robust to a wide range of possible distortions. Others are general frameworks for obtaining robustness to specific distortions.

In Section 9.2, we analyze the effects of some *valumetric* distortions (i.e., distortions that change the values of individual pixels or audio samples). Common valumetric distortions that occur to photographs, music, and video can often be modeled as combinations of additive noise, amplitude changes, linear filtering, and/or quantization. *Temporal* and *geometric* distortions (such as delay, translation, rotation, and scaling) are discussed in Section 9.3.

9.1 APPROACHES

There are several broad strategies for making watermarks detectable after their cover Works are distorted. Some of these aim to make watermarks robust to all possible distortions that preserve the value of the cover Work. Others are strategies for handling specific types of distortions. In practice, a watermarking system will employ multiple strategies for handling various types of distortion. For example, image watermarking systems commonly use *redundant embedding* (Section 9.1.1) to handle cropping, filtering, and addition of band-limited noise, but use *inversion at the detector* (Section 9.1.5) to handle geometric distortions.

Sections 9.1.1 through 9.1.4 describe strategies for ensuring that watermarks are unmodified, or only slightly modified, by normal processing. These rely on finding transform domains in which some terms are likely to be unaffected by processing.

In contrast, Sections 9.1.5 and 9.1.6 describe methods based on inverting the effect of distortions. In these, we allow the watermark to be modified by the distortion, but we counteract that modification in either the detector (Section 9.1.5) or embedder (Section 9.1.6). These types of strategies are often used to handle temporal and geometric distortions (see Section 9.3).

A note on terminology: In Sections 9.1.1 through 9.1.4 we talk about embedding watermarks "in" certain coefficients. For example, a watermark might be embedded in the low-frequency Fourier coefficients. This means that the reference watermark has nonzero values in the low frequencies of its Fourier representation, but zeros (or small magnitudes) in the high frequencies. It does not necessarily mean that the computation of watermark embedding is performed in the Fourier domain. A low-frequency reference pattern can be just as easily represented and embedded in some other domain, such as the spatial or temporal

domain. In other words, the description and analysis of a watermark may be performed in one domain, while embedding and detection are implemented in another.

9.1.1 Redundant Embedding

Often, when a Work is distorted not all coefficients in its representation are affected equally. As a trivial example, consider an image cropped along its right side. In a pixel representation of this image, only the rightmost pixels are affected; the remaining pixels are untouched. A more interesting example occurs when a Work is convolved with a band-pass filter. If the Work is represented in the spatial or temporal domain, this filtering is likely to change all samples. However, in the Fourier domain the filter affects only the frequencies outside the passband.

One general strategy for surviving a wide variety of distortions is to embed a watermark redundantly across several coefficients. If some of these coefficients are damaged, the watermark in other coefficients should remain detectable. A simple example of this approach is the spatial tiling used in the E_BLK_BLIND/D_BLK_CC system (System 3). This tiling provides robustness against spatially varying distortions, such as cropping.

Watermarks that are embedded redundantly by tiling can be detected in a number of ways. One method is to combine data from all the tiles in a Work and then decode the result. This is the method employed in the D_BLK_CC detector, which adds all the tiles together and tests for the watermark in their average.

Another method is to test for the watermark in each tile independently, announcing that the mark is present if it is detected in more than some fraction of the locations. This approach is employed in several proposed systems, such as Bas *et al.* [32].

Yet another method is to test for the watermark in each tile, and then combine the resulting detection values. For example, we could compute the correlation coefficient between the reference vector and each 8×8 block in an image, and then compare the sum of all the results against a threshold [91]. Note that if we used linear correlation instead of the correlation coefficient, this would be the same as computing the correlation between the entire image and a tiled reference pattern, such as that shown in Figure 8.20.

Although the most obvious examples of redundant embedding involve tiling, other approaches are possible. In general, we can say that a watermark is embedded redundantly, in some domain, if the watermark can be detected in several subsets of the coefficients. For example, consider a white-noise watermark embedded in an image with the E_BLIND blind embedder. Although this is not a tiled pattern, the watermark is redundantly embedded in the spatial domain because it is detectable from several subimages (assuming perfect registration between the subimage and the reference pattern).

The idea of redundant embedding can be taken a step further by embedding a message into a Work multiple times, using different watermark encodings, where each encoding is designed to survive a distinct set of distortions. Because the Work is unlikely to undergo all distortions to which the marks are susceptible, at least one of the marks is likely to survive. This idea is explored in [272].

9.1.2 Spread Spectrum Coding

When applied in the frequency domain, the idea of redundant embedding leads to the well-known communications paradigm of spread spectrum coding. In a spread spectrum communications system, messages are encoded with sequences of symbols (see Chapter 4). The symbols are transmitted in a temporal sequence, each one being represented by a signal referred to as a *chip*. Typically, chips are pseudo-random sequences of 1s and 0s. In the frequency domain, they are spread across a wide range of frequencies. Thus, if the signal is distorted by some process that damages only a fraction of the frequencies, such as a band-pass filter or addition of band-limited noise, the chips will still be identifiable.

Figure 9.1 illustrates a simple spread spectrum audio watermarking system, similar to that proposed in Provos [337]. At the embedder, we begin with a message, m, which is error encoded and denoted m_c. Let m_c be represented by a sequence of symbols drawn from a reasonably large alphabet, such as the 128 ASCII characters. Each message symbol is then spread spectrum modulated into a 512-bit pseudo-random sequence (or chip). The resulting sequence of chips is perceptually shaped, and then added to the audio cover Work. At the receiver, the reverse process takes place. First, the perceptual shaping is (approximately) inverted. Next, each chip is identified by correlating against the 128 chips in the alphabet, and choosing the one with the best correlation. This produces a sequence of error-coded symbols that are then decoded to produce the message, m.

Spread spectrum communications have two characteristics that are important to watermarking. First, the signal energy inserted into any one frequency is very small. This very low signal-to-noise ratio reduces the risk of perceptible artifacts. Note, however, that despite the low signal-to-noise ratio present in any

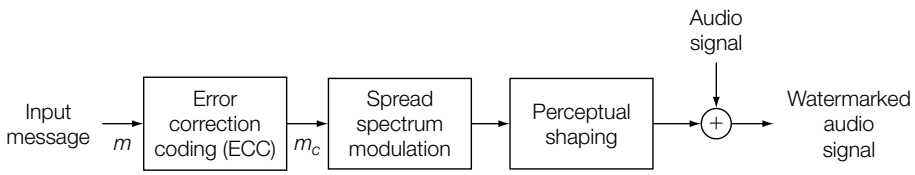


FIGURE 9.1

Simple spread spectrum audio watermarking method.

single frequency the detector output signal may still have a high signal-to-noise ratio as it despreads or concentrates the energy present in a large number of frequencies. Second, the fact that the watermark is dispersed over a large number of frequencies also provides robustness to many common signal distortions.

9.1.3 Embedding in Perceptually Significant Coefficients

In practice, it is not always appropriate to embed a watermark in *all* coefficients of a Work's representation. Some coefficients are so susceptible to distortions that they are unlikely to be useful. For example, most forms of image-processing damage the high frequencies in an image's Fourier representation. That is, lossy compression quantizes many high frequencies to zero, halftoning adds high-frequency noise, scanning applies a low-pass filter before sampling, and so on. It is almost certain, therefore, that any watermark information placed in high frequencies will be damaged in a processed image. If the information in such unreliable coefficients is used during detection, it will likely reduce the reliability of the system.

Furthermore, fidelity constraints may mean that any watermark energy embedded in unreliable coefficients must be offset by a reduction of energy in more reliable coefficients. Thus, even if watermark data in unreliable coefficients were useful during detection, its cost may outweigh its benefit. It is better to put as much of the watermark as possible into reliable coefficients.

How do we decide which coefficients are reliable and which are not? One very general answer, proposed in Cox *et al.* [96], is that coefficients that are *perceptually significant* are likely to be reliable, whereas those that are *perceptually insignificant* are likely to be unreliable. An ideal perceptually significant coefficient is one that never changes unless the Work is perceptibly distorted. In general, all of the normal compression, transmission, and display technologies applied to a Work are specifically designed to preserve perceptually significant features.

Conversely, a watermark need not necessarily survive a process that damages perceptually significant coefficients. By definition, damaging these coefficients degrades the perceptible quality of a Work. If the resulting Work is so badly distorted that its value is lost, the information embedded in it may no longer be useful. For example, an audio clip that has been processed to the point of being unrecognizable is not protected by copyright law [416]. Therefore, a copyright notice embedded as a watermark would not be meaningful.[2]

[2] This is a peculiar characteristic of watermarking, as opposed to other forms of data hiding. As we have defined it, a watermark carries information that pertains to the cover Work. Therefore, the watermark usually becomes irrelevant if the cover Work is unrecognizable or worthless. This contrasts with, for example, an application of steganography for covert communications,

A major problem with embedding in perceptually significant coefficients is that it is directly opposed to the objective of creating imperceptible watermarks. To make a watermark imperceptible, we might want to put most of its energy into perceptually *insignificant* coefficients. This, in fact, is one of the effects of the example perceptual shaping algorithms described in the previous chapter, which generally move image watermark energy into high frequencies. Thus, the objective of fidelity is often fundamentally opposed to the objective of robustness.

The solution to this problem employed in Cox *et al.* [96] is to spread the watermark over a large number of perceptually significant coefficients in the frequency domain. Because spread spectrum coding allows a very small energy to be used in each frequency, the change in each coefficient can be small enough to be imperceptible.

An alternative solution is to embed in coefficients that have medium levels of perceptual significance. That is, their perceptual significance is neither so high that we cannot embed a watermark in them imperceptibly nor so low that they are unlikely to survive normal processing. For example, many image watermarking techniques embed in images' middle frequencies, avoiding the lowest frequencies because they are too perceptible, and avoiding the highest frequencies because they are too unreliable [202, 324, 325].

9.1.4 Embedding in Coefficients of Known Robustness

When we embed in perceptually significant coefficients, we are attempting to make a watermark that will survive all conceivable processes that preserve a Work's value. However, in many applications we are not concerned with all conceivable processes, but with a specific set of processes that might occur between embedding and detection. In such cases, we can deal with these processes more directly.

First, the watermark should be described in a domain that is likely to be robust to the processes of interest. For example, if we are more concerned with having an image watermark survive spatial shifting than we are with having it survive linear filtering, we might choose to embed in the image's Fourier magnitudes. In some cases, it might be best to use other domains, such as the log-polar Fourier transform [262, 317] (discussed in Section 9.3) or the cepstrum transform [35, 44, 315].

Once we have chosen a representation in which to describe our watermark, we can identify the coefficients that best survive the expected distortions. For some distortions, this can be done analytically. For others, it can be done by empirical test. The tests are straightforward and involve comparing the content directly after embedding and directly before detection.

---

where the hidden message is unrelated to the cover Work and is likely to retain its value even if the cover Work is completely lost.

By comparing corresponding coefficients, it is possible to determine how the channel between the embedder and the detector affects each coefficient. Such experiments need to be performed over a wide variety of content, and numerous trials are often needed to build a satisfactory model with sufficient statistical reliability.

For example, consider a video watermarking system that must survive recording on VHS tape. If the watermark is to be described in the block DCT domain, we need to identify the coefficients of an 8×8 DCT that are least affected by this recording. This can be accomplished by recording a large number of frames onto the tape, and then redigitizing them and comparing their block DCT coefficients.

In the above example, we perform a statistical test over many Works, and then determine a fixed set of coefficients to use in the watermark. However, a given coefficient might behave differently in different Works. Consider the problem of making watermarks survive adaptive compression. Adaptive compression algorithms, such as those in [208, 318], examine the Works being compressed and adjust the amount of quantization applied to each coefficient. Consequently, a given coefficient might be heavily quantized in one Work and quantized very little in another. This suggests that a watermark should be embedded into a different set of coefficients for each Work.

One technique for tailoring the choice of coefficients to individual Works is to measure the relative robustness of each coefficient just prior to embedding a watermark. This can be done by applying several simulated distortions to the Work and measuring their effects on the representation of that Work in the chosen domain. The watermark is then embedded into the coefficients found to be most robust, which might be a different set of coefficients for each Work. The list of coefficients used is provided to the detector along with the (possibly distorted) marked Work. Such a system has been investigated in Liang and Rodriguez [258]. Of course, this technique requires an informed detector.

9.1.5 Inverting Distortions in the Detector

The previously discussed approaches attempt to create a watermark that remains relatively unchanged after normal processing. A fundamentally different approach is to attempt, during the detection process, to invert any processing that has been applied since the watermark was embedded.

Many processes can be either exactly or approximately inverted. For example, if an audio clip is delayed by an integral number of samples, the detector can exactly invert the process by simply advancing the clip by the same number of samples (if we ignore the effect of cropping). A clockwise rotation of an image can be inverted by a counterclockwise rotation of the same angle. For rotations that are not multiples of 90 degrees, this inversion is likely to be approximate because of problems with interpolation and round-off error. If a detector can determine that one of these processes has been applied to a Work between the time of embedding and the time of detection, it can

apply the inverse process to the Work to obtain a close approximation of its unprocessed version.

Alternatively, depending on the watermarking algorithm, it may be possible for a detector to apply a distortion to the reference watermark, rather than applying the inverse distortion to the Work. For example, consider an image watermark detector that computes the linear correlation between the received Work and a reference pattern, as in the D_LC algorithm of System 1. If such a detector determines that the Work has probably been filtered by a low-pass filter, it can detect the mark by applying the same filter to the reference pattern before computing the correlation. This might be preferred over applying the inverse process to the received Work for reasons of computational cost, or because the inverse process is imprecise or unstable.

The difficult step in inverting distortions in the detector is determining what distortion to invert. In the worst case, the detector might have to apply an exhaustive search, testing for a watermark after inverting each of a large number of possible processes. For example, an image watermark detector might try to detect after rotating the image 0 degrees through 359 degrees in increments of 1 degree.

For some types of processing, it is possible to reduce the cost of search and false positive probability. If an algorithm can identify one or a small number of candidate distortions that are likely to have taken place, the detector need test for watermarks after inverting only those distortions. Such an algorithm must be specifically defined for each class of distortions.

The problem of identifying the distortion is generally easier in an informed detector than in a blind detector, because an informed detector has the possibility of comparing the received Work against the original. When the correct inverse process is applied to the received Work, it should match the original (with the exception of its watermark, if it has one). For many types of processes, fast algorithms exist to find the parameters that lead to the best match between two Works. The approach of inverting distortions at the detector is most commonly applied for temporal and geometric distortions, as discussed in Section 9.3.

9.1.6 Preinverting Distortions in the Embedder

In some circumstances, there may be a small set of very well-defined distortions a watermark must survive. For example, there are a small number of well-defined formats for video signals, and a given video signal might be converted from one to another. For illustrative purposes, we focus on the way DVD video players handle the *aspect ratio,* or relative width and height, of the picture.[3] Standard NTSC and PAL televisions have an aspect ratio of 4:3 (i.e., the television is 4 units wide and 3 units high). On the other hand, high-definition television (HDTV)

[3] The reader is directed to Taylor [402] for more details on how DVD handles aspect ratio.

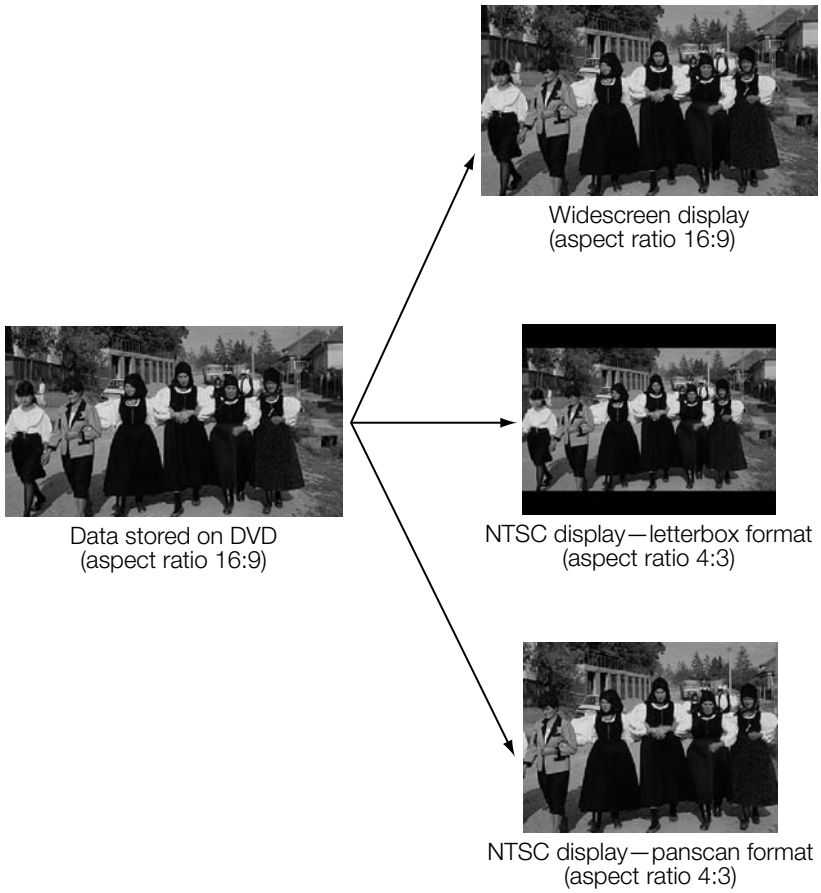


FIGURE 9.2

Three display modes of DVD.

has an aspect ratio of 16:9. To support both formats, DVD disks store the full 16:9 image, and offer two methods of squeezing the picture onto 4:3 standard television screens. In *letterbox* format, the picture is reduced in size so that it fits onto the 4:3 screen in its entirety, with some black lines added above and below. In *panscan* format, the picture fills the 4:3 screen, but is cropped at the left and right. These formats are illustrated in Figure 9.2.

Watermarks for DVD video should be embedded in the widescreen (16:9) source. Consequently, the watermark would be detected when displayed on a widescreen. However, both letterbox and panscan modes introduce distortions that may cause the watermark to go undetected. Thus, we have a small number (two) of well-defined geometric distortions the watermarked Work is likely to undergo.

To combat each of these distortions, we can embed a special watermark in the original format. This watermark is predistorted such that after conversion to the corresponding display mode (panscan or letterbox) the watermark will be detected. The basic idea is to embed more than one watermark in the content. The first watermark is not predistorted and will be detected by the watermark detector when no format conversions occur. A second watermark is predistorted for letterbox. A third is predistorted for panscan. Depending on the display mode, only one of the three watermarks will be detected.

Multiple watermarks can be inserted in content either by embedding one on top of another or by time multiplexing. In the latter case, one frame of a video sequence might contain the first watermark, the second frame the next watermark, and so on. Or the multiplexing may be coarser, with the first watermark being embedded in the first 10 seconds of video, the next watermark in the next 10 seconds, and so on. The disadvantage of such an approach is that for a given format the corresponding watermark is only detectable $1/N$ of the time, where N is the number of predistorted watermarks. The alternative of overlaying these multiple watermarks avoids this problem. However, each watermark must be embedded with, on average, $1/N$ the power if fidelity constraints are to be conserved.

The simplest procedure for embedding a predistorted watermark consists of three steps:

1. Apply the expected distortion to the Work being watermarked (e.g., conversion to letterbox or panscan mode).
2. Embed the watermark in the distorted Work. At this point, the watermark embedder is unaware of the distortion. It simply embeds a watermark in the same manner as for undistorted content.
3. Apply the inverse distortion to the watermarked, distorted Work.

Note that this procedure can be applied with any watermark embedding algorithm.

Unfortunately, the previously outlined procedure can introduce unacceptable fidelity loss when applied with a distortion that is not perfectly invertible. In both letterbox and panscan conversion, some information is lost, and therefore the inverse distortion of the last step must introduce some distortion in image quality. For this reason, it may be preferable to apply a slightly more complicated procedure, which works for certain types of distortions [94].

1. Make a copy of the Work being watermarked and apply the expected distortion to it, obtaining a distorted copy of the Work, $\mathbf{c_d}$.
2. Create a watermarked version of the distorted copy, $\mathbf{c_{dw}}$.
3. Find the pattern that was added to the distorted copy, $\mathbf{c_d}$, in Step 2: $\mathbf{w_{da}} = \mathbf{c_{dw}} - \mathbf{c_d}$.

4. Perform the inverse distortion on the added pattern, $\mathbf{w_{da}}$, to yield the corresponding pattern, $\mathbf{w_a}$, for the original image. For example, if the transformation to be compensated for was "letterbox" mode, $\mathbf{w_a}$ would be obtained by vertically expanding $\mathbf{w_{da}}$.
5. Finally, add $\mathbf{w_a}$ to the original cover Work, $\mathbf{c_o}$, to obtain the watermarked Work, $\mathbf{c_w}$.

These steps are illustrated in Figure 9.3.

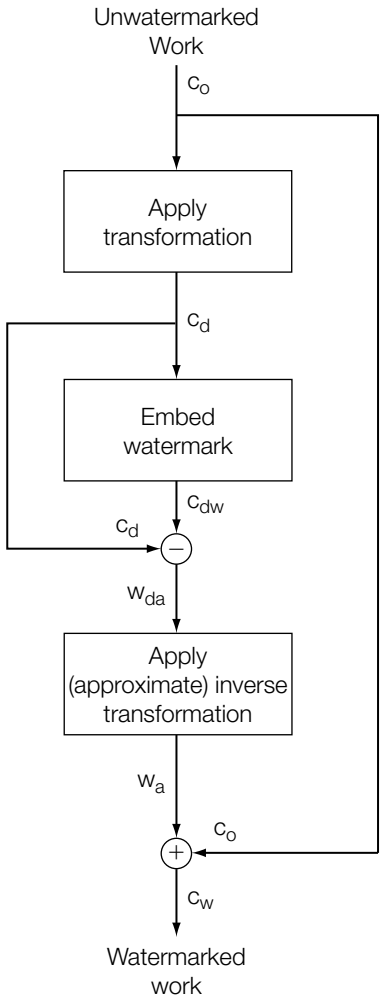


FIGURE 9.3

The embedding process for inserting a predistorted watermark.

When the distortion is later applied to the watermarked Work, $\mathbf{c_w}$, the result will be approximately the same as the watermarked, distorted Work, $\mathbf{c_{dw}}$, created in step 2. Thus, the watermark will be detected by the same procedure designed to detect a normal watermark.

In the rare circumstance in which we can identify a small number of specific distortions the watermark must survive, predistorting watermarks at the embedder can have several advantages. First, it is very general, being applicable to any watermarking system for almost any distortion, as long as the distortion is approximately invertible. Second, it adds no cost to the detector, which is important in applications for which detector cost must be minimized. Finally, it can be implemented after detectors are deployed. This can be important if a new distortion is introduced into normal content handling (e.g., if a new format conversion is introduced into the DVD standard).

9.2 ROBUSTNESS TO VALUMETRIC DISTORTIONS

We now examine the effects of four major types of valumetric distortion on watermark detection: additive noise, amplitude changes, linear filtering, and lossy compression. Most of these can be countered with simple applications of the previously discussed techniques.

9.2.1 Additive Noise

Some processes that might be applied to a Work have the effect of adding a random signal. That is,

$$\mathbf{c_n} = \mathbf{c} + \mathbf{n}, \tag{9.1}$$

where $\mathbf{c}$ is a Work and $\mathbf{n}$ is a random vector chosen from some distribution, independently of $\mathbf{c}$. For example, audio broadcast over a radio channel might be corrupted by white-noise, resulting in a hiss or static. Similarly, video broadcast over a television channel might pick up video snow. In these cases, the noise is independent of the Work. Such noise processes are cases of *additive noise*.

Because additive noise is independent of the Work, it is easy to address analytically. For this reason, most analysis of watermarking algorithms assumes that Works are transmitted over additive noise channels. For example, almost all the discussion of robustness in Chapters 3 through 5 addresses only additive noise. As a result, most robust watermarking algorithms are specifically designed to survive this type of distortion. In the following investigation we examine the effects of additive white-noise on two of our example watermarking systems.

INVESTIGATION

Effects of Additive Noise

In the case of watermarking systems that employ linear correlation, the detection statistic is known to be optimal in the presence of additive white Gaussian noise. Thus, we can reasonably expect that the addition of such noise will have little effect on detection.

In the case of watermarking systems that employ normalized correlation, the detection statistic is not optimal for additive noise. However, in Chapter 5, normalized correlation's weakness against this type of distortion is explicitly handled by the design of a *robustness measure,* R^2, that estimates the amount of noise a given watermark can survive. The E_BLK_FIXED_R (System 8) embedding algorithm is then designed to ensure that all watermarks are embedded with a specific robustness to additive noise. Thus, we expect the addition of noise to this system to have little effect on detection as long as the noise is small compared to that specified by the R^2 value.

Experiment 1

Two thousand images were watermarked by the E_FIXED_LC fixed linear correlation embedder (see System 2). We used this embedder, rather than the simpler E_BLIND embedder, because it minimizes the variance in detection values for undistorted images, and this makes it easier to see the effect subsequent distortion has on detection variance. The images were embedded with a threshold value of $\tau_{lc} = 0.7$, strength parameter of $\beta = 0.3$, and message $m = 1$, giving a target linear correlation of 1. Each image was then distorted 10 times with additive white-noise of 16 different powers, and the detection values were measured by the D_LC detector. Figure 9.4 shows the result of this robustness experiment. The points on the graph show the mean detection values obtained as a function of noise amplitude. The error bars show the standard deviations of those detection values (mean plus or minus one standard deviation).

The mean detection value shown in Figure 9.4 should be unaffected by noise, in that the addition of any noise pattern has as much chance of increasing linear correlation with the watermark as it has of decreasing it. In fact, the mean detection value remains high, but it does decrease slightly as the amplitude of the noise increases. This decrease is not directly due to the added noise but to the fact that the pixel values are rounded and clipped to 8-bit values after the noise is added.

Unlike the means of the detection values, the standard deviations increase as noise amplitude increases. As a result, an increasing fraction of images should fall below the threshold. However, this does not become a serious problem within the

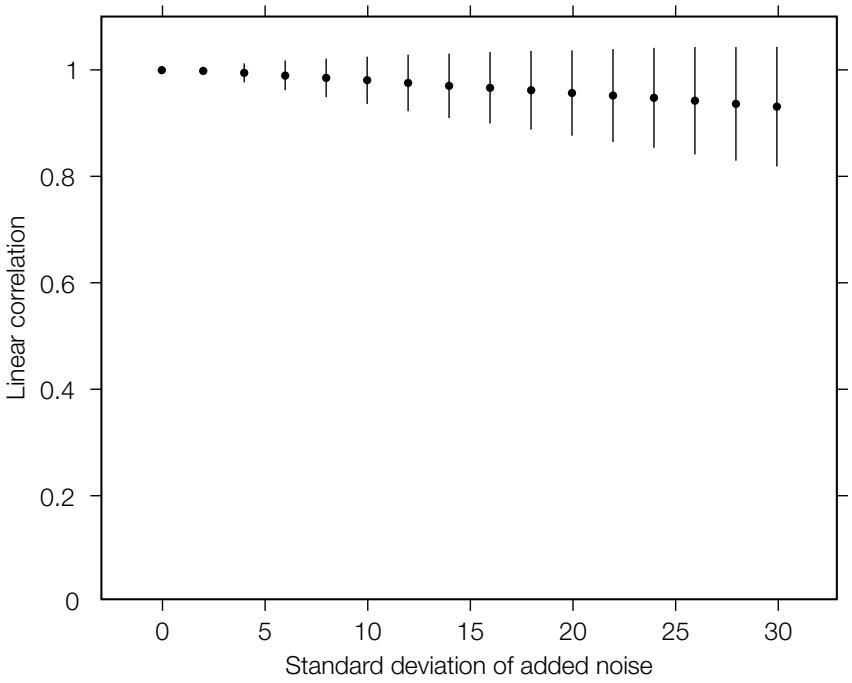


FIGURE 9.4

Results of additive white Gaussian noise test using E_FIXED_LC/D_LC watermarking system. The points show the mean detection values obtained. The error bars indicate the standard deviations of detection values (mean value plus or minus one standard deviation).

range of noise added during the experiment, and, with a detection threshold of $\tau_{lc} = 0.7$, the watermark is successfully detected in more than 97% of the images, even at the highest noise amplitude tested. At the maximum noise amplitude the image is so severely distorted that it might no longer be of interest.

Experiment 2

Two thousand images were watermarked by the E_BLK_FIXED_R embedder (see System 8), with a detection threshold of $\tau_{nc} = 0.55$ and a target robustness value of $R^2 = 30$. Each image was distorted 10 times with each of 16 different powers of noise, and the detection values were measured by the D_BLK_CC detector. Figure 9.5 shows the results of this experiment.

This graph shows that the addition of noise affects the mean normalized correlation more seriously than it affects linear correlation. However, for more than 85% of the images the detection values still remain above the threshold of $\tau_{nc} = 0.55$ after adding noise with a standard deviation of 18. Figure 9.6 shows an example of an image with this much noise added.

Note that in Figure 9.5 the variance of detection values in undistorted images is larger than that for the E_FIXED_LC/D_LC system. This is because the E_BLK_FIXED_R embedder is holding the robustness constant, but allows the detection value to vary (see Chapter 5 for more on this issue). Up to a point, as the amount of noise increases, the detection value variance actually *decreases* slightly. This reflects the nonlinear nature of normalized correlation. At values close to 1 or −1, normalized correlation is much more sensitive to small changes in the extracted vector than it is at values close to 0.

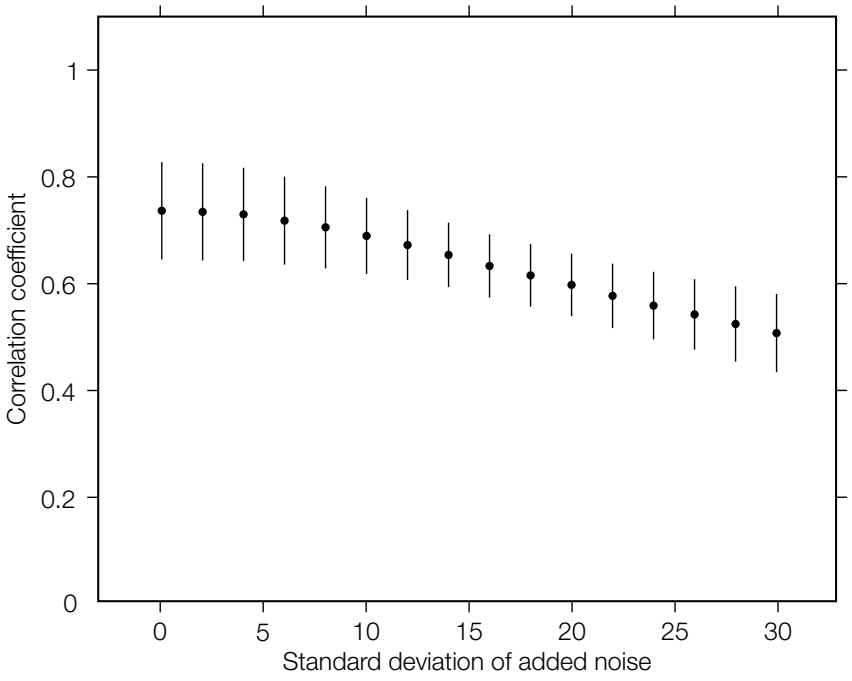


FIGURE 9.5

Results of additive white Gaussian noise test using the E_BLK_FIXED_R/D_BLK_CC watermarking system. The points show the mean detection values obtained. The error bars indicate the standard deviations of detection values (mean value plus or minus one standard deviation).

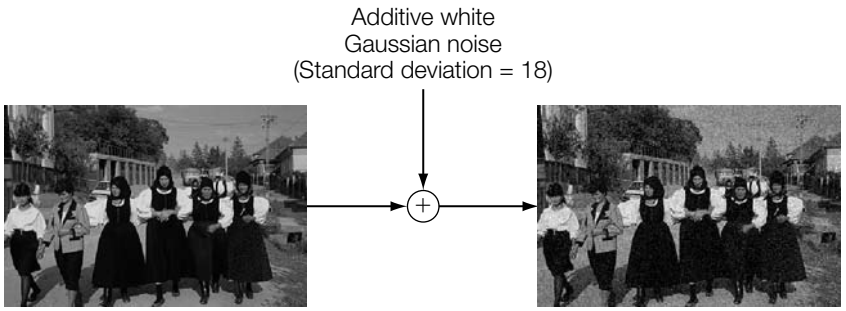


FIGURE 9.6

Visual effect of adding noise with a standard deviation of 18.

9.2.2 Amplitude Changes

In theoretical discussions of watermarking systems, it is tempting to deal with only additive noise, because this form of distortion is easy to analyze. However, in reality, many, if not most, processes applied to watermarked Works are not well modeled by additive noise. The change in a Work is usually correlated with the Work itself. In fact, many processes are *deterministic* functions of the Work. A simple, but important, example is that of changes in amplitude. That is,

$$\mathbf{c_n} = \nu\mathbf{c}, \tag{9.2}$$

where $\mathbf{c}$ is a Work and ν is a scaling factor. For music, this simply represents a change of volume. In images and video, it represents a change in brightness and contrast.

If a watermark is to be fully robust to amplitude changes, the detection region in media space must be a set of rays emanating from the origin. That is, if $\mathbf{c_w}$ is a watermarked Work, the detection region must include all points along the ray from the origin of media space (i.e., pure black or pure silence) through $\mathbf{c_w}$. Of course, linear correlation does not have this property, in that it changes linearly with amplitude. Normalized correlation, on the other hand, is specifically designed to be independent of amplitude.

INVESTIGATION

Effects of Amplitude Change

In this investigation we examine the effects of amplitude in two of our example watermarking systems. We look first at a system that uses linear correlation detection, in which we expect detection values to vary with changes in amplitude.

Then we examine a normalized correlation system in which we expect to find that the scaling has little, if any, effect on the detection value.

Experiment 1

The E_FIXED_LC/D_LC linear correlation system from System 2 was used to embed watermarks in 2,000 images. The contrast of each of these watermarked images was then modified by scaling the amplitude of the image by several scaling factors between $\nu = 1$ and $\nu = 0$. The resulting detection values are plotted in Figure 9.7, with the scale factor decreasing from left to right, so that the severity of the attack increases from left to right. This figure shows, not surprisingly, that linear correlation falls off linearly with decreasing scale.

Experiment 2

The experiment was then repeated with the E_BLK_FIXED_R/D_BLK_CC watermarking system from System 8, in which the detection statistic is a correlation coefficient. Again, 2,000 images were watermarked and the amplitudes scaled. Figure 9.8 shows the detection results, again plotted with the scale factor

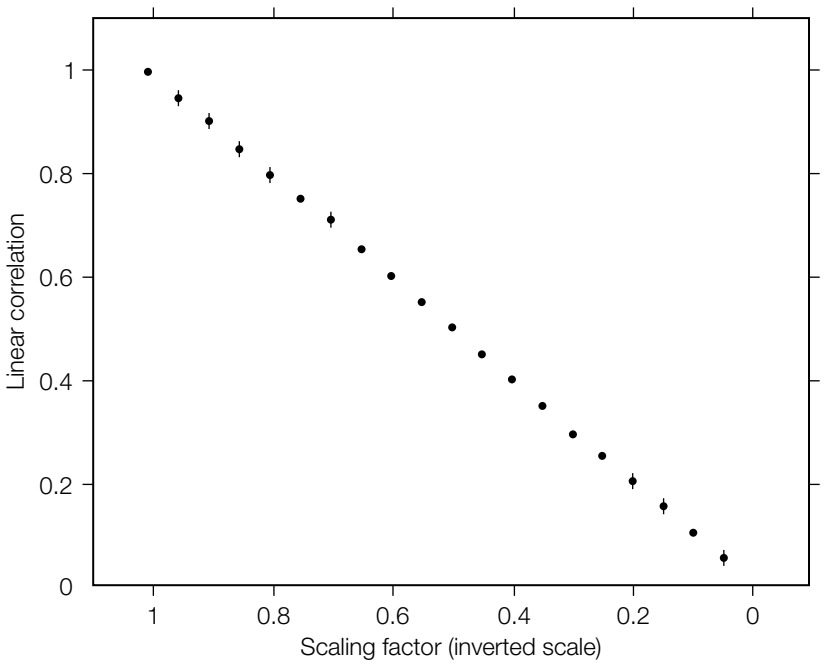


FIGURE 9.7

Results of contrast scaling test using E_FIXED_LC/D_LC watermarking system. The points show the mean detection values obtained. The error bars indicate the standard deviations of detection values.

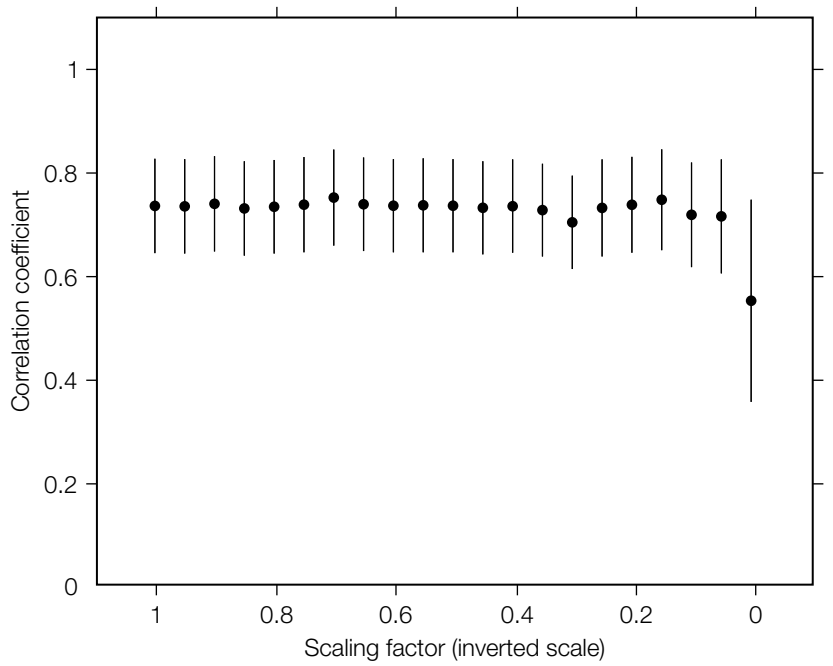


FIGURE 9.8

Results of contrast scaling test using the E_BLK_FIXED_R/D_BLK_CC watermarking system. The points show the mean detection values obtained. The error bars indicate the standard deviations of detection values.

decreasing from left to right, so that the severity of the attack increases from left to right. These results show that normalized correlation remains fairly constant with scale, except that at very low scaling factors detection becomes much more noisy, and the mean begins to decline. This is caused by the quantization of the scaled pixel intensities to 8-bit values.

9.2.3 Linear Filtering

Another common type of signal processing that changes Works in a deterministic fashion is linear filtering. That is,

$$\mathbf{c_n} = \mathbf{c} * \mathbf{f}, \tag{9.3}$$

where $\mathbf{c}$ is a Work, $\mathbf{f}$ is a filter, and $*$ denotes convolution. Many normal operations on images and audio are explicitly implemented with linear filters. The blurring and sharpening effects in image editing programs apply simple

filtering operations, as do base and treble adjustments in a stereo system. In addition, many lossy processes, although not explicitly implemented with filters, can be modeled by them. For example, distortions due to VHS recording can be modeled by applying a low-pass filter to each scan line of video. Playback of audio over loudspeakers can also be approximated as a filtering operation.

If a filter is symmetric about its center in the temporal or spatial domain, all coefficients in its Fourier representation are real. Because convolution in the temporal or spatial domains corresponds to multiplication in the Fourier domain, filtering with a symmetric filter is equivalent to scaling each element of a Work's Fourier representation by a real-valued scalar. Thus, we can think of each frequency as being either attenuated or amplified by a symmetric filter.

Application of an asymmetric filter affects not only the amplitudes but the phases of the frequencies in a Work's Fourier representation. Like the change in amplitude, the change in phase is frequency dependent. The phases of some frequencies might be changed substantially, whereas others are left unaffected.

The effects of a filter on watermark detection depend on how much energy the added watermark pattern has in each frequency. In general, changes in frequencies where the reference pattern has high energy have a greater effect on detection than do changes in frequencies where the reference pattern has little or no energy. Thus, to make a watermark robust to a known group of filters that might be applied to a Work, we should design the reference pattern to have most of its energy in the frequencies the filters change the least. This idea is examined in the following investigation.

An alternative way to make watermarks survive filtering is to design a watermark extraction method, or detection statistic, that is invariant to convolution with symmetric filters. Note that such convolution affects only the magnitude of each Fourier coefficient and has no effect on its phase. Thus, if only the phase is used during detection, the watermark will be unaffected by this filtering. This idea has been employed in at least two watermarking systems [216, 316].

INVESTIGATION

Effects of Low-Pass Filtering

In this investigation, we test the hypothesis that the effects of low-pass filtering depend on the frequency content of the reference pattern. Robustness to low-pass distortion is measured for two reference patterns that have different spectral characteristics.

Experiment 1

The E_FIXED_LC embedder from System 2 was used to embed a Gaussian, white-noise pattern in 2,000 images. Because we expect this pattern to be fragile

against low-pass filtering, we used a higher embedding strength, $\beta = 1.3$, than in most of our experiments. With a detection threshold of $\tau_{lc} = 0.7$, this gave a target correlation of 2.

The watermarked images were then distorted with Gaussian, low-pass filters of varying standard deviation (width). The filters were computed as

$$f[x, y] = \frac{f_o}{\sum_{x,y} f_o[x, y]}, \tag{9.4}$$

where

$$f_o[x, y] = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{9.5}$$

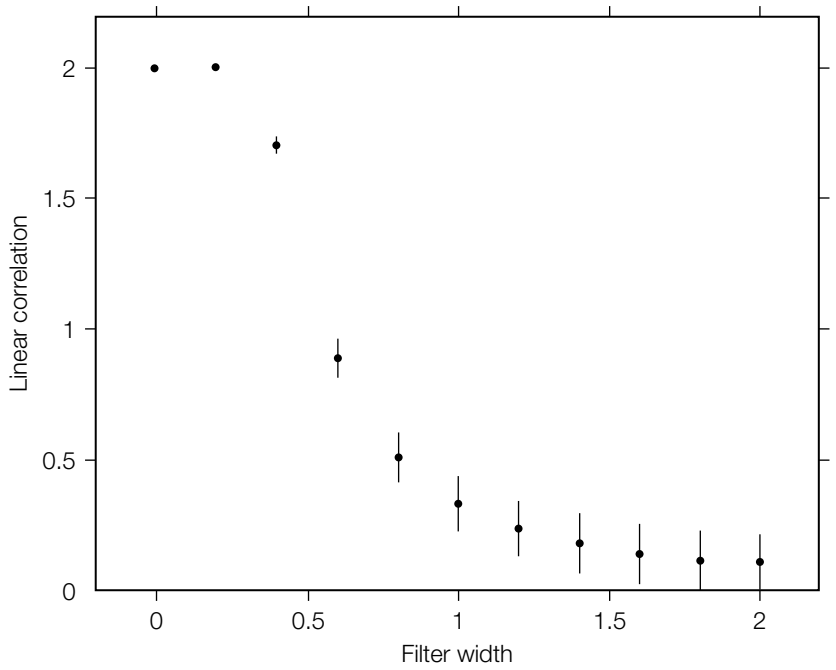


FIGURE 9.9

Results of low-pass filtering test using the E_FIXED_LC/D_LC watermarking system and a white-noise watermark reference pattern. The points show the mean detection values obtained. The error bars indicate the standard deviations of detection values.

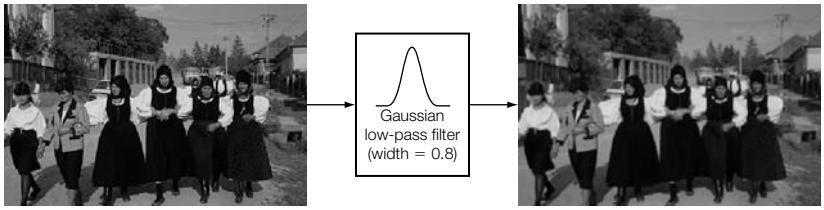


FIGURE 9.10

Visual effect of convolving an image with a Gaussian low-pass filter with a width of 0.8.

and the range of x, y included all large values of $f_o[x, y]$. As the width, σ, of the filter increased, more distortion was introduced.

The D_LC detector was then applied, and the resulting detection values are shown in Figure 9.9. This graph indicates that the white-noise pattern is extremely sensitive to low-pass filtering. The detection values begin falling off sharply when the filter width exceeds 0.4, and the watermark goes largely undetected when the filter width reaches only 0.8. Figure 9.10 shows that the visual impact of filtering with a Gaussian filter with a width of 0.8 is not terribly significant.

Experiment 2

Next, we modified the reference pattern and repeated the experiment. A band-limited reference pattern was created by filtering high and low frequencies out of the white-noise pattern. The low frequencies were filtered out because they are difficult to embed with high fidelity. The high frequencies were filtered out because they are most affected by low-pass filters. Figure 9.11 shows the resulting reference pattern. Such midfrequency reference patterns are used in many watermarking algorithms (see, for example, [202, 324, 325]).

This pattern was embedded into 2,000 images again using the E_FIXED_LC embedder, with $\tau = 0.7$, $\beta = 1.3$, and the resulting images were filtered as before. The detection values that resulted from application of the D_LC detector are shown in Figure 9.12. Because correlation with this midfrequency reference pattern is independent of the highest frequencies, its detection essentially ignores the aspects of the image that are most damaged by low-pass filtering, and the detection values fall off much more slowly. The watermark is still detected in about 85% of the images after filtering with a filter of width 2. Figure 9.13 shows that the visual impact of such a filter is quite noticeable. In some applications, we would not require the watermark to survive more serious distortions.

Of course, as discussed in Chapters 3 and 7, different reference patterns can have wildly different false positive and fidelity behaviors in the E_FIXED_LC/D_LC watermarking system. In a practical implementation of a midfrequency.

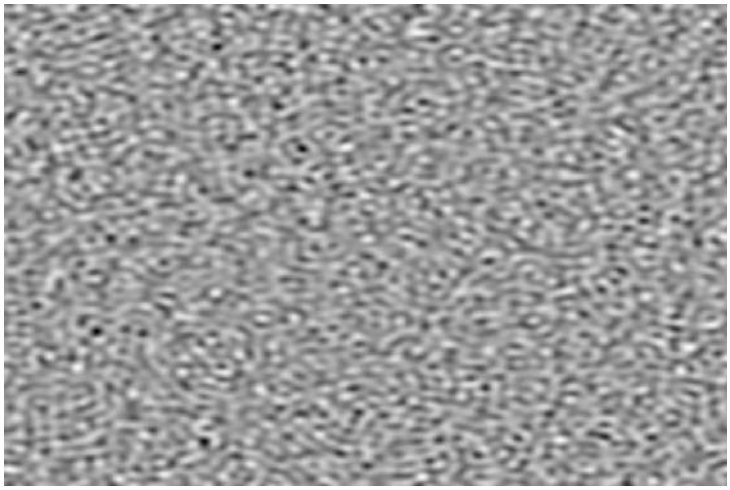


FIGURE 9.11

Midfrequency watermark reference pattern created by filtering highest and lowest frequencies out of a white-noise pattern.

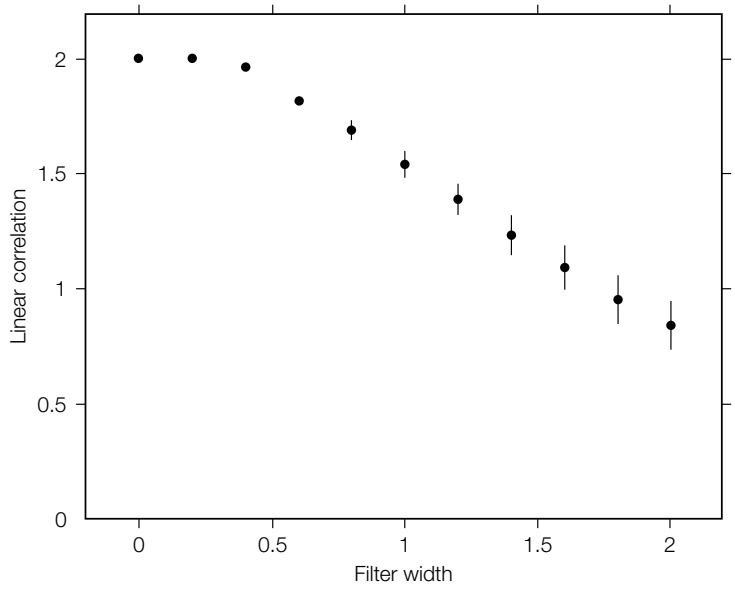


FIGURE 9.12

Results of low-pass filtering test using the E_FIXED_LC/D_LC watermarking system and a midfrequency watermark reference pattern. The points show the mean detection values obtained. The error bars indicate the standard deviations of detection values.

watermark, care must be taken to ensure that the false positive rate and the fidelity impact are not too high.

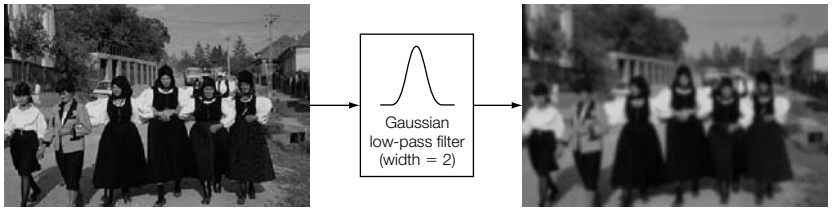


FIGURE 9.13

Visual effect of convolving an image with a Gaussian low-pass filter with a width of 2.

9.2.4 **Lossy Compression**

In lossy compression, a Work is represented in such a way that when the Work is decompressed the result is *not an identical* copy of the original. The loss of information can be acceptable because the computer representation of the signal contains redundancy with respect to what is needed for human perception.

Many people have recognized that there is a fundamental conflict between watermarking and lossy compression. An ideal lossy compressor should quantize all perceptually equivalent Works into a single compressed representation. No two compressed representations should result in perceptually identical Works. If this is not the case, then the lossy compression algorithm is not removing all of the redundancy present in the Work. Now, clearly, if a watermark is to survive lossy compression, the compressed representations of the original and watermarked Works must be different. However, if this is true, our ideal model of lossy compression requires that the original and watermarked Works must be perceptually distinct. In other words, it should not be possible to have a watermark survive lossy compression without affecting the fidelity of the Work.

An interesting illustration of the conflict is provided in Zhu and Tewfik [470]. If we are given a lossy compression algorithm and a watermark that survives the compression process, we can use the data payload available to the watermark to encode a portion of the compressed representation. In so doing, we improve the compression algorithm and remove the redundancy the watermarking algorithm was originally exploiting. Admittedly, this is likely to provide only a small gain in compression rate. Fortunately, in practice, lossy compression algorithms are far from ideal, and it is still reasonably straightforward for a watermarking algorithm to survive lossy compression while maintaining excellent fidelity.

9.2.5 Quantization

In this section, we examine the effects of quantization. Quantization can be modeled as

$$\mathbf{c_n}[i] = q \left\lfloor \frac{\mathbf{c}[i]}{q} + 0.5 \right\rfloor, \tag{9.6}$$

where q is a constant *quantization factor,* and $\lfloor x + 0.5 \rfloor$ rounds x to the nearest integer. A minor form of this distortion occurs as part of most watermark embedding processes, when the watermarked Work is quantized to integer values ($q = 1$) before being stored. We have seen the effect of this quantization in Figure 9.8, where it made watermarks undetectable in many images that were scaled to nearly 0 brightness. However, the most serious quantization usually takes place during lossy compression.

Most common lossy compression algorithms follow the basic structure shown in Figure 9.14. First, a linear, energy-preserving transform, such as the DCT or wavelet transform, is applied to the Work being compressed. Next, each of the resulting terms is independently quantized, in the manner shown in Equation 9.6. Finally, some combination of entropy coding techniques (such as Huffman coding, run-length coding, or predictive coding) is applied to represent the quantized Work with the smallest number of bits practical. Because the transform and entropy coding steps are lossless, the watermark need only survive the quantization step. It is for this reason we devote the rest of this section to discussing the effects of quantization.

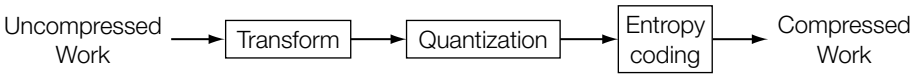


FIGURE 9.14

General form of many lossy compression algorithms.

Characteristics of Quantization Noise

It is often tempting to assume that quantization is equivalent to the addition of independent noise, which was discussed in Section 9.2.1. The amount of error quantization that adds to any given term of the Work depends on where that term lies within the range between two integral multiples of the quantization factor, q. If it lies exactly on a multiple of q, quantization will have no effect on it, and the error will be 0. If it lies exactly between two multiples of q, it will be either increased or decreased by $q/2$. At first glance, it would seem that each term might lie anywhere within the range between the two closest multiples of q, with equal probability, so that the noise added to it is drawn from a uniform distribution between $-q/2$ and $q/2$. This would imply that quantization should have little effect on linear correlation.

In fact, the assumption of independent quantization noise is reasonably accurate when q is relatively small. Widrow [445] pointed out that the maximum value of q that leads to independent noise can be found by applying a version of Nyquist's sampling theorem. The sampling theorem states that a continuous signal can be reconstructed from a sampled version if the sampling frequency is at least twice the highest frequency present in that signal. Widrow showed that if the "quantization frequency," $\phi = 2\pi/q$, is at least twice as high as the highest frequency present in the Fourier transform of a random variable's PDF, that PDF can be reconstructed from the quantized PDF. More importantly, he showed that under these conditions the resulting quantization noise is independent of the random variable.

Unfortunately, in watermarking we have little control over either the quantization factor or the PDF of the unwatermarked content, and Widrow's condition frequently is not satisfied. If it were satisfied, we should expect linear correlation to be robust to quantization errors, in that it is robust against additive noise. However, the following investigation shows that linear correlation can be quite sensitive to quantization.

INVESTIGATION

Effects of Simulated JPEG Compression

In this Investigation, we simulate the effects of JPEG compression by applying quantization in the block-DCT domain. We examine the effect of such quantization on watermarks embedded by the E_FIXED_LC embedder and detected with the D_LC detector.

Experiment

A white-noise watermark was embedded in each of 2,000 images using the E_FIXED_LC embedder with an expected threshold of $\tau_{lc} = 0.7$ and a "strength" of $\beta = 0.3$. To quantize in the block-DCT domain, each watermarked image was divided into 8×8 pixel blocks and the two-dimensional DCT of each block was computed. The DCT coefficient values were then quantized to integer multiples of a quantization factor, q, using a different value for q for each of the 64 terms of a block. Finally, the inverse DCT was applied.

The quantization factors were obtained by multiplying a global quantization level, Q, by the matrix of term-specific quantization factors shown in Table 9.1. This is the luminance quantization matrix used in the JPEG lossy compression algorithm [322]. Thus, for example, when Q was 2, the DC term of each 8×8 block was quantized with a quantization factor of $q = 32$, and the highest-frequency term was quantized with $q = 198$.

The results of applying the D_LC detector to the watermarked, quantized images are shown in Figure 9.15. The x-axis of the figure indicates different values of Q (with $Q = 0$ indicating no quantization). The y-axis indicates the resulting detection values. Clearly, quantization can have a significant effect on linear correlation.

Table 9.1 Luminance quantization matrix used in JPEG. The upper left value (16) is the base quantization factor for the DC term of each 8 × 8 block. The lower right value (99) is the base quantization factor for the highest-frequency terms. These base values are multiplied by a global quantization value (Q) to obtain the actual quantization factor used.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

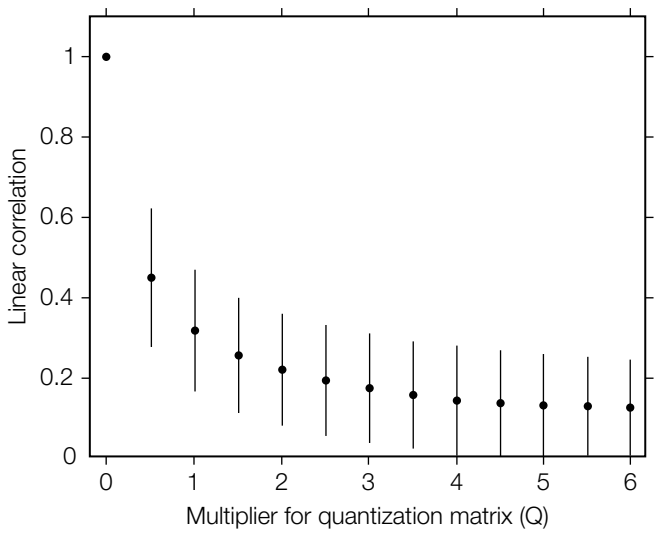


FIGURE 9.15

Effect of quantization on linear correlation detection. Two thousand images were watermarked with white-noise reference marks, using the E_FIXED_LC embedding algorithm ($\tau_{lc} = 0.7$, $\beta = 0.3$). They were then converted to the block DCT domain, and the DCT coefficients were quantized by quantization factors computed as $Q \times q[i]$, where Q is a global multiplier (x-axis in this graph), and $q[i]$ is a frequency-dependent quantization value (see Table 9.1). Points show mean detection values. Bars show plus and minus one standard deviation.

Analytic Model of Quantization Noise on Linear Correlation

To obtain a more accurate model of quantization's effect on linear correlation watermark detection, it is useful to view the watermarking and quantization process, together, as a case of *dithered quantization*. This is a modified quantization process sometimes used in communications to control the effect of quantization noise.

A dither signal, $\mathbf{d}$, is added to the original signal, $\mathbf{x}$, prior to quantization. This dither signal is usually a pseudo-random signal. After quantization, the signal is commonly transmitted to one or more receivers that then attempt to reconstruct the signal. In *subtractive dither,* the dither signal is subtracted from the quantized signal in order to reconstruct the signal, as shown in Figure 9.16. In *nonsubtractive dither,* the receiver does not have access to the dither signal. Schuchman [363] showed that if a well-chosen dither signal is added to the input signal prior to quantization, the quantization error will be independent of the input signal, $\mathbf{x}$.

If the dither signal, $\mathbf{d}$, in Figure 9.16 is replaced by the unwatermarked cover Work, $\mathbf{c_o}$, and the transmitted signal, $\mathbf{x}$, is replaced by the watermark reference pattern, $\mathbf{w_r}$, the result is exactly a classical blind embedder. Subtractive dither is equivalent to informed detection, in which the original Work is subtracted from the received Work, and nonsubtractive dither is equivalent to blind detection. Several authors have observed these equivalences [82, 126, 127].

Eggers and Girod [127] have used the results from Schuchman to analyze how the watermark signal is affected by quantization.[4] They argue that the effect of quantization on watermark detection is determined by the expected correlation between the quantization noise and the watermark reference pattern, $E(\mathbf{n}\mathbf{w_r})$. This is the same as the expected product of any element of $\mathbf{n}$ and any element of $\mathbf{w_r}$; that is, $E(nw)$, where n is any element of $\mathbf{n}$ and w is any element of $\mathbf{w_r}$.

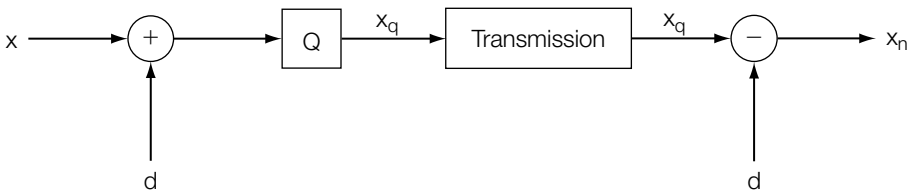


FIGURE 9.16

Subtractive dithered quantization.

[4] Eggers and Girod chose to consider the watermark as the dither signal. Because we are interested in detection of the watermark signal rather than the cover Work, we consider the watermark to be the signal and the content to be the dither.

The analysis of Eggers and Girod [127], along with the necessary background from Schuchman [363], is presented in Section B.5 of Appendix B. To apply this analysis, we first need to specify the probability density functions for the watermark and the content. The probability distribution for the watermark is up to the designer of the watermarking system. For convenience, we will assume it is Gaussian; that is,

$$P_W(x) = \frac{1}{\sigma_W\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma_W^2}}. \tag{9.7}$$

As discussed in Chapter 3, the probability distribution for the content is difficult to model accurately. In Chapter 7, when deriving the whitening filter for D_WHITE (System 9), we assumed an elliptical Gaussian distribution as a model of image pixel values. In that case, the Gaussian assumption led to a reasonable result (i.e., the resulting whitening filter works). However, Eggers and Girod investigated the use of an elliptical Gaussian model for the distribution of coefficients in an image's block DCT and found that the resulting predictions of $E(nw)$ did not match experimental results. Instead, they suggested using a Laplacian distribution, as recommended in [41, 345], or a generalized Gaussian, as recommended in [41, 305]. The generalized Gaussian yielded better results, but it was not possible to compute $E(nw)$ analytically. We therefore conclude this section by showing the equation for $E(nw)$ when the content, $\mathbf{c}$, is drawn from a Laplacian distribution with zero mean and standard deviation $\sigma_{\mathbf{c}}$:

$$P_c(x) = \frac{1}{\sqrt{2}\sigma_{\mathbf{c}}} e^{-\frac{\sqrt{2}}{\sigma_{\mathbf{c}}}|x|}. \tag{9.8}$$

Under these assumptions, the expected value, $E(nw)$, is given by

$$E(nw) = \sigma_W^2 \sum_{b=-\infty}^{\infty} (-1)^b \frac{1}{1 + 2(\pi b \sigma_c/q)^2} e^{-2(\pi b \sigma_W/q)^2}. \tag{9.9}$$

Figure 9.17 shows the results of a simple, numerical experiment, verifying that Equation 9.9 accurately predicts $E(nw)$. One hundred thousand values for $\mathbf{c}$ and w were drawn at random from the distributions described by Equations 9.7 and 9.8, with $\sigma_{\mathbf{c}} = 2$ and $\sigma_w = 0.2$. For each of several quantization factors, 100,000 values of n were computed from the values of $\mathbf{c}$ and w, and the average value of nw was found. The results are plotted as points in the figure. The solid line plots the expected values of nw, as computed by Equation 9.9. Note that when q is very large the expected value of nw tends toward $-\sigma_w^2 = -0.04$. Thus, for $q/\sigma_{\mathbf{c}} > 0.5$ the quantization noise is independent of the watermark signal. However, for values less than this there is a negative correlation that diminishes the output from a linear correlator. Eggers and Girod have examined the usefulness of this result in predicting the effects of JPEG compression [127].

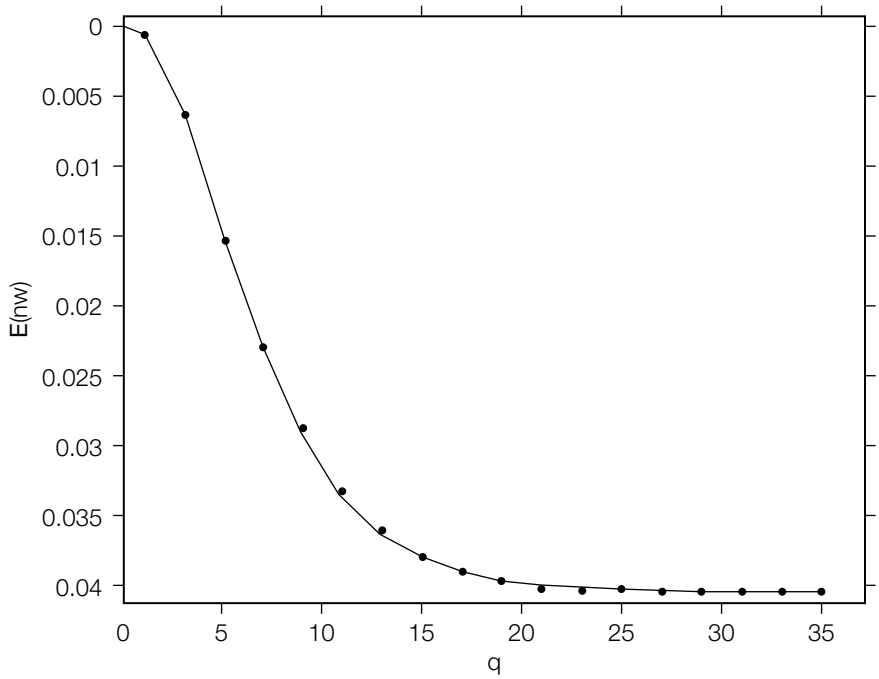


FIGURE 9.17

Normalized expected value of the detection value as a function of the quantization step size, q. The solid line represents the predicted value, whereas the points represent the results of a simulation.

9.3 ROBUSTNESS TO TEMPORAL AND GEOMETRIC DISTORTIONS

Whereas the previous section examined the traditional sources of noise in a communications system, this section addresses temporal and geometric distortions. Temporal distortion, affecting audio and video signals, includes delay and temporal scaling. Geometric distortion affecting image and video data includes rotation, spatial scaling, translation, skew or shear, perspective transformation, and changes in aspect ratio. In both the temporal and geometric cases the distortion can be global, affecting all samples similarly, or may vary locally.

Although many different approaches have been investigated [118], robustness to temporal and geometric distortion remains one of the most difficult outstanding areas of watermarking research. Most suggested approaches fall into one of the following categories: exhaustive search, synchronization or

registration, autocorrelation, invariant watermarks, and implicit synchronization. We examine each of these techniques in turn, but first describe the temporal and geometric distortions themselves.

9.3.1 Temporal and Geometric Distortions

Temporal scaling in audio commonly occurs as a result of simple speed changes in tape players. In video, temporal scaling often occurs in conversions between the frame rates of different television standards. In both audio and video, shifts can occur whenever a watermark detector is presented with signals taken from an arbitrary point in the audio or video signal stream.

Audio scaling may also result from the application of sophisticated processes designed to change the duration of sounds without changing the wavelengths of their component sinusoids [115, 133, 172, 250]. However, these more sophisticated forms of scaling are not discussed in this chapter.

A Work that has been subjected to a time scaling of s and then a delay of δ can be written as

$$\mathbf{c_n}[t] = \mathbf{c}[st + \delta]. \tag{9.10}$$

Geometric distortions to video commonly occur when video undergoes format changes, as in the case illustrated in Figure 9.2, and when standard-definition video is upsampled to high-definition formats. Geometric distortions to photographs commonly occur through a variety of user manipulations, some intentional and others unintentional. Often, printing, photocopying, and scanning will introduce a variety of unintended geometric distortions.

In two-dimensional visual data there are two, potentially different, scaling parameters and two different translation parameters. When the vertical and horizontal scaling factors differ, there will be a change in the *aspect ratio*. Additionally, a visual Work can undergo rotation and shear (or skew). All of these geometric distortions can be expressed as an affine transformation given by

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \begin{bmatrix} x_t \\ y_t \end{bmatrix}, \tag{9.11}$$

where (x_0, y_0) represents the undistorted location of each pixel and (x_n, y_n) represents the distorted location. Whereas the translation is completely represented by the vector (x_t, y_t), the two-by-two matrix is used to define all other affine transformations. For example, scaling can be described by the matrix

$$\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}, \tag{9.12}$$

with change in aspect ratio occurring when $s_x \neq s_y$. Rotation by an angle, θ, can be implemented with the matrix

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{9.13}$$

Shear in the x-dimension and y-dimension can be respectively described by the matrices

$$\begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \tag{9.14}$$

and

$$\begin{bmatrix} 1 & 0 \\ b & 1 \end{bmatrix}. \tag{9.15}$$

In addition to these affine distortions, other common geometric distortions include reflection, perspective distortion, and geometric warping.

Different watermark patterns have different levels of *natural robustness* to temporal and geometric distortions. In the one-dimensional case, the linear correlation between a delayed, time-scaled Work and the reference mark, $\mathbf{w_r}$, will only be high if $\mathbf{w_r}[t] \cdot \mathbf{w_r}[st + \delta]$ is high. Thus, the correlation between a reference mark and a delayed, time-scaled version of itself defines its natural robustness to delays and scales. For sufficiently small values of s and δ, many one-dimensional reference marks will show some natural robustness. Low-pass signals exhibit some natural robustness to small shifts. However, the autocorrelation of a truly white-noise signal drops to zero at a shift of only one sample. Similarly, different two-dimensional reference patterns exhibit varying degrees of natural robustness to shift, rotation, scaling, skew, and so on [64, 65].

9.3.2 Exhaustive Search

The simplest approach for watermark detection after a temporal or geometric distortion is an exhaustive search. After defining a range of likely values for each distortion parameter and a search resolution for each, every combination of parameters is examined. The search range can often be constrained by the assumption that if any distortions were applied they have had minimal impact on the perceptual quality of the Work. The resolution of the search can be determined by the natural robustness of the reference pattern to the distortions of interest.

Each combination of distortion parameter values represents a hypothetical distortion that might have been applied to the Work after the watermark had been embedded. The search can involve inverting each hypothetical distortion and then applying the watermark detector once for each possible reference pattern. Alternatively, we can apply the hypothetical distortion to the reference pattern prior to detection.

When considering an exhaustive search, two primary issues arise, both related to the large number of applications of the watermark detector. The first, and perhaps most obvious, concern is computational cost. The amount of computation necessary increases with the size of the search space.

Consider an audio source sampled at 14.1 kHz in which a watermark is embedded once each second. If we allow for simple temporal scaling of ±3%, an exhaustive search might examine all temporal scalings from 97% to 103% in 0.1% increments, and temporal delays of ±0.5 seconds in increments of 1 sample. This search would require about $N = 860{,}000$ detection operations.

In two-dimensional Works, the size of the search space can increase dramatically. For example, consider an exhaustive search that examines all rotations from −179 to +180 degrees in 1-degree increments, horizontal and vertical scalings from 50% to 200% in 1% increments, and vertical and horizontal translations of ±100 pixels in increments of 1 pixel. This search would require about $N = 330$ *billion* detection operations.

The second important issue that arises with multiple applications of a watermark detector is the effect on the false positive probability. For each unwatermarked Work, we test N distorted versions in the detector. If the watermark is found in at least one of these versions, the detector will produce a false positive. Denoting the random-Work false positive probability of any single version of a Work by P_{fp}, the probability that at least one of the N versions will cause a false positive is bounded by $N \times P_{fp}$. When N is large, this can be unacceptable.

Computation and false positive probability are two practical forces that limit the size of the search space. Thus, effective uses of exhaustive search rely on techniques that result in small searches [179]. For example, in an image marking space constructed by summing all 8×8 blocks (as in the E_BLK_BLIND and D_BLK_CC algorithms), all image translations can be roughly modeled as circular shifts of the 8×8 extracted vector. Thus, a full translational search, with one pixel resolution in both vertical and horizontal dimensions, would require only 64 applications of the detector.

9.3.3 Synchronization/Registration in Blind Detectors

Both the computational cost and the increase in false positive probability associated with an exhaustive search can be avoided if the Work suspected of containing a watermark can be aligned with the reference pattern prior to a single application of the watermark detector. This process is called *synchronization* in audio literature and *registration* in image-processing literature. When the original Work is available at the detector, techniques from the pattern recognition literature can be used to align the original Work with the Work suspected of containing a watermark [57, 251, 276, 390, 425, 457].

A common synchronization approach for blind detectors is the embedding of a dedicated synchronization pattern in addition to the payload-carrying

reference patterns [103, 323, 409]. Because the synchronization pattern is known, the detector can employ one of many well-known registration techniques. As with all watermark patterns, synchronization patterns are designed to have very low power compared to that of the Work. To ease the registration task in such a "noisy" environment, the synchronization pattern can be specially designed for easy identification in the face of temporal and geometric distortions [347].

With this approach, watermark detection involves first finding the registration pattern in a suspect Work. The temporal/geometric distortion that had been applied to that Work then can be identified by comparison of the extracted registration pattern and the embedded registration pattern. That distortion is inverted and detection of the data-carrying watermark proceeds.

Systems that rely on synchronization patterns to correct for temporal and/or geometric distortions have two failure modes. As we have previously discussed, a false negative can occur because of problems with the payload-carrying watermark. It may not have been effectively embedded or it may have been distorted, intentionally or unintentionally, between the time of embedding and detection. The second potential cause of a false negative is problems with the synchronization pattern. A correct positive detection requires that both the payload-carrying marks and the synchronization pattern be successfully embedded and successfully detected.

The use of a synchronization pattern also has negative security implications. Typically, the same synchronization pattern is used for many different Works. This eases the task of the detector in finding a distorted synchronization pattern, but it may also allow the synchronization pattern to be discovered from a set of watermarked Works. Once the synchronization pattern is discovered by an adversary, it can be removed, thus limiting the ability of the watermarking scheme to counter the effects of temporal and geometric distortions. It is therefore important that the synchronization pattern itself be as secure as the data-carrying watermark. (See Chapter 10 for a detailed discussion of security.)

If the synchronization pattern is added along with the data-carrying watermark, the fidelity of the resulting Work is likely to decrease. Alternatively, the amplitude of these marks can be reduced to maintain the fidelity, but this will likely reduce the overall robustness of the system to additive noise, filtering, and so on. Thus, the use of synchronization patterns to provide robustness to temporal and geometric distortions comes at a cost to either fidelity or general robustness.

9.3.4 **Autocorrelation**

In some cases, the embedded pattern can serve both as the synchronization pattern and as the payload-carrying pattern. Typically, this requires that the data-carrying watermark have some properties that allow for synchronization [347]. In the *autocorrelation* approach, this property is periodicity.

The autocorrelation of a Work typically has a large peak at zero (corresponding to the signal energy) and then decays rapidly at nonzero shifts. This is even more true when examining the autocorrelation of a "white" or uniformly distributed signal. When a periodic, white synchronization pattern is embedded in a Work, the resulting autocorrelation will contain a periodic train of peaks identifying periodicity of the added pattern in the Work. This, in turn, can be used to identify and invert any scaling applied to the Work since the embedding of the synchronization pattern [198, 199, 243].

Autocorrelation methods have significant potential. However, similar to the approach of adding a synchronization mark, they have two failure modes. For successful detection, both the identification of the temporal/geometric distortion and the detection of the watermark after inversion of that distortion must be successful. Depending on the application, both of these processes may need to be robust and/or secure.

9.3.5 Invariant Watermarks

Rather than detect and invert the temporal and geometric distortions, an alternative approach is the design of watermarks that are invariant to such distortions. The Work is projected to an invariant feature vector as part of the signal extraction process.

Consider the following derivation of a one-dimensional feature vector that is invariant to delay and scaling of the temporal axes. It starts with a commonly used feature vector that is invariant to shifts or delays, the Fourier magnitude [48]. Next, note that scaling of a Work's time axis appears as a scaling of the frequency axis in the Fourier domain, so that in one dimension we can write

$$|\mathcal{F}\{\mathbf{c}(st-\delta)\}| = \frac{1}{s}|\mathbf{C}(f/s)|. \tag{9.16}$$

This scaling of the frequency axis can be treated as a shift along a log axis by defining the signal, $\mathbf{C}'$, as follows:

$$\mathbf{C}'(u) \equiv \mathbf{C}(e^u), \tag{9.17}$$

so that

$$\mathbf{C}'(\log f) = \mathbf{C}(f). \tag{9.18}$$

Equation 9.16 then can be written as

$$|\mathcal{F}\{\mathbf{c}(st-\delta)\}| = \frac{1}{s}|\mathbf{C}'(\log f - \log s)|. \tag{9.19}$$

Therefore, in this domain, temporal delay has no effect, and temporal scaling is seen as a coordinate shift and a change in amplitude. By applying a second

Fourier magnitude we can ensure that the only effect of temporal scaling and delay is a change in amplitude; that is,

$$|\mathcal{F}\{|\mathcal{F}\{\mathbf{C}(st-\delta)\}|\}| = \left|\mathcal{F}\left\{\frac{1}{s}|\mathbf{C}'(\log f - \log s)|\right\}\right| \tag{9.20}$$

$$= \frac{1}{s}|\mathcal{F}\{|\mathbf{C}'(\log f)|\}|. \tag{9.21}$$

We can then use a detection statistic that is invariant to amplitude changes, such as normalized correlation. This is an example of a timescale and time-delay invariant representation of a one-dimensional signal.

These ideas have been extended to two dimensions to build feature vectors that are invariant to rotation, scaling, and translation. The trick is to represent the Fourier image in polar coordinates and then apply a logarithmic transformation to the radial axis. This can be implemented with a Fourier-Mellin transform [203, 265, 368, 369, 455]. Watermarking methods have been proposed that use these techniques as part of the watermark extraction process [262, 317]. A similar approach has been taken to generate watermarks that are invariant to changes in aspect ratio [263].

9.3.6 Implicit Synchronization

There is a class of blind detection watermarking techniques in which the suspect Work goes through a synchronization process prior to detection. However, rather than a synchronization pattern, the actual features of the Work are used. The watermark is embedded at a time or geometry relative to the features of the original Work. We refer to this type of synchronization as *implicit synchronization* because the synchronization pattern is implied by the Work itself.

A simple application of this approach is that of Wu *et al.* [452], in which a watermark signal is embedded in an audio stream just after the detection of "salient points." In this example, salient points are defined as locations at which the signal is climbing fast to a peak value. Such an approach will provide robustness to delay because the location of the watermark remains constant relative to the salient points.

Similarly, [31, 32, 113, 351] are examples in which feature points are extracted from an image. The reference patterns representing the watermark are then warped to fit the geometries implied by those points.

In another approach, the marking space is defined as a canonical, normalized space based on the geometric image moments [17]. Based on these moments, an image is transformed into a form that is independent of its scale or orientation. This form is also invariant to horizontal and vertical reflection. The marking is applied in this space, and then the inverse transformation restores the original reflection, orientation, and scale. During detection, the moments are again calculated and used to estimate the normalization parameters. Once the image is normalized, the watermark can be detected.

Implicit synchronization requires that the salient features be reliably extracted during detection. Some distortions may affect the locations of salient features relative to the Work. When these distortions are applied after watermark embedding but before detection, the implicit synchronization can fail and the watermark can go undetected.

9.4 SUMMARY

The robustness of a watermarking method is a measure of the ability of the watermark to survive legitimate and everyday usage of the content. General methods for achieving high robustness were presented along with specific methods for handling some of the most common types of processing. The main points of this chapter were the following.

- We discussed several general approaches to making watermarks robust:
  - *Redundant embedding* can increase robustness to cropping, filtering, and additive noise. The redundancy can be in the sample domain, the frequency domain, or any other domain in which only part of the signal is distorted by processing.
  - *Spread spectrum* watermark reference patterns are redundant in the spatial and frequency domains. These provide general robustness against filtering, additive noise, and cropping.
  - Embedding watermarks in perceptually significant components of content ensures their robustness against any processing that maintains acceptable fidelity.
  - It is sometimes possible to explicitly identify components of content that are robust to expected processes and embed the watermark in these. If the detector is informed, this may be done on a Work-by-Work basis.
  - If a detector can determine that a specific process has been applied to a Work since the time it was watermarked, the detector might either invert that process, or apply it to the reference mark.
  - Sometimes we can anticipate that watermarks will be subject to one of a small collection of possible distortions. In these cases, it may be possible to apply the inverse distortion during the embedding process.
- Valumetric distortions (as opposed to geometric distortions) can often be modeled as combinations of additive noise, amplitude changes, linear filtering, and lossy compression.
- Linear correlation detection is specifically designed for robustness to additive noise. The use of a robustness metric in the embedder can also provide robustness to additive noise in a normalized correlation detector.

- The use of normalized correlation provides robustness to amplitude changes.
- The use of midfrequency reference patterns provides robustness to low-pass filtering.
- In theory, lossy compression is diametrically opposed to watermarking, since compression attempts to remove all redundancy from content, and watermarks attempt to code information in that redundancy. In practice, however, lossy compression algorithms are far from perfect, and it is possible to make watermarks that survive them.
- Most lossy compression algorithms involve quantizing a Work in some transform domain.
- The noise introduced by quantization is not always well modeled as independent, additive noise. We presented a method for predicting the impact of quantization on linear correlation given a distribution of unwatermarked content, a watermark distribution, and a quantization step size.
- Geometric distortions, such as temporal delay or spatial scaling, are generally more difficult to handle than valumetric distortions, and robustness against them is a current topic of research. We presented several broad approaches to this problem:
  - *Exhaustive search* entails inverting a large number of possible distortions, and testing for a watermark after each one. As the number of possible distortions increases, the computational cost and false positive probability using this approach can become unacceptable.
  - Synchronization/registration patterns can be embedded in content to simplify the search. These prevent an increase in the false alarm rate and are usually more computationally efficient than an exhaustive search. However, they introduce two failure modes: failure to correctly detect the registration pattern and failure to detect the watermark after registration.
  - In the autocorrelation approach, we embed a periodic watermark and register based on the peaks in a Work's autocorrelation pattern.
  - *Invariant watermarks* can be constructed using such techniques as log-polar Fourier transforms. These remain unchanged under certain geometric distortions, thereby eliminating the need to identify the specific distortions that have occurred.
  - In *implicit synchronization*, we register according to feature points found in the original, unwatermarked Work. This depends on development of a reliable feature-extraction method.

This page intentionally left blank

CHAPTER 10

Watermark Security

The security of a watermark refers to its ability to resist intentional tampering, as opposed to the common signal distortions discussed in Chapter 9. Intentional tampering or hostile attack becomes an issue when someone is motivated to prevent a watermark from serving its intended purpose. Watermarks can be attacked in a variety of different ways, and each application requires its own type of security.

We begin, in Section 10.1, with a discussion of the types and levels of security that might be required of a watermarking application. We then, in Section 10.2, discuss the relationship between watermarking and cryptography. We in find that there are cryptographic tools that can be directly applied to watermarking systems to achieve some types of security. Those security areas in which cryptographic tools are helpful are distinguished from those that must be addressed in other ways. Finally, in Section 10.3, we describe a number of well-known attacks on watermarking systems. For some of these attacks, reliable countermeasures are known, but security against some others remains a topic of research.

10.1 SECURITY REQUIREMENTS

The security requirements for a watermarking system vary greatly from application to application. In some applications watermarks need not be secure at all, because there is no motivation to tamper with or circumvent the watermark's intended pupose. For example, most of the device-control applications described in Section 2.1.7 use watermarks only to add value to the content in which they are embedded. In such cases, no one benefits from tampering, and therefore the watermarks need not be secure against any form of malicious attack, although they still need to be robust against normal processing.

Those applications that do require security often must defend against very different forms of attacks. In some cases, the types of security required can even vary between different implementations of the same application. For example, a copy-control system (Chapter 2) might use watermarks to identify copyrighted

Works that people are not allowed to copy. Such watermarks should be secure against unauthorized removal. Alternatively, a copy-control system could use watermarks to identify Works that people *are* allowed to copy. The absence of a watermark indicates that copying is not allowed. With such a design, an adversary is not motivated to remove watermarks but to *embed* watermarks. Thus, a watermarking algorithm for this second copy-control system must be designed to prevent unauthorized embedding.

Furthermore, the *level* of security required by different applications can vary, depending on the level of sophistication of expected adversaries. Any military application of watermarks would require the highest levels of security possible, because the adversaries must be assumed to have all of the resources of an enemy nation at their disposal. At the other extreme, an application meant to restrict small children's access to certain Works may need to be secure only against the simplest attacks, if any.

In this section, we first discuss some broad restrictions that various applications place on manipulations of their watermarks. Specifically, in a given application, certain people may be restricted from embedding, detecting, and/or removing watermarks. Within each of these categories of restriction there are subtle variations on the types of attack an adversary may attempt. For example, an adversary restricted from detecting marks may attempt to fully detect and decode the watermark message, or may attempt only to determine that the watermark is present, without reading the message it contains. These variations are the topic of Section 10.1.3. Finally, we discuss the assumptions to be made about the adversary when judging the security of a watermark.

10.1.1 Restricting Watermark Operations

In every application of watermarking, some people must have the ability to embed, detect, and/or remove watermarks, whereas others must be restricted from performing some, or all, of these actions. Secure watermarks are required to enforce these restrictions. Consider the following three scenarios.[1]

- *Scenario 1:* Alice is an advertiser who embeds a watermark in each of her radio commercials before distributing them to 600 radio stations. She then monitors the radio stations with a watermark detector, logging the broadcast of her commercials and later matching her logs with the 600

[1] Those familiar with the literature on cryptography will recognize the names of Alice and Bob in these scenarios. These two, along with Carol and eavesdropping Eve, are standard characters in descriptions of cryptography problems. We use them here for the same reason they are used in cryptography: because they are more interesting than simply A, B, and C. However, by using these characters, we do not mean to imply that the problems we are describing are necessarily cryptographic problems or that they can be addressed with cryptographic solutions. The relationship between cryptography and watermarking is addressed in Section 10.2.

invoices she receives, thus identifying any false charges. Bob operates one of these 600 radio stations and would like to air one of his own commercials in place of one of Alice's. However, he still wants to charge Alice for the air time. Therefore, Bob secretly embeds Alice's watermark into his own advertisement and airs it in place of Alice's commercial. Alice detects this watermark and is fooled into believing that her advertisement was correctly broadcast.

- *Scenario 2:* Alice owns a watermarking service that, for a nominal fee, adds an owner identification watermark to images that will be accessed through the Internet. Alice also provides an expensive reporting service to inform her customers of all instances of their watermarked images found on the Web. Her customers use this information to identify any unauthorized distribution of their images. Bob builds his own web crawler that can detect watermarks embedded by Alice, and offers a cheaper, competing reporting service, luring away Alice's reporting service customers. Bob can offer a cheaper service because he does not incur the cost associated with embedding watermarks.
- *Scenario 3:* Alice owns a movie studio, and she embeds a copy-control watermark in her movies before they are distributed. She trusts that all digital recorders capable of copying these movies contain watermark detectors and will refuse to copy her movie. Bob is a video pirate who has a device designed to remove the copy protection watermark from Alice's movies. Using this device, Bob is able to make illegal copies of Alice's movies.

In each of these scenarios, Bob manages to circumvent the intended purpose of Alice's watermarks by performing an operation he is not authorized to perform. In Scenario 1, he performs an *unauthorized embedding* operation, or *forgery attack,* by embedding a watermark that only Alice should be able to embed. In Scenario 2, he performs an *unauthorized detection* operation, or *passive attack,* by detecting watermarks that only Alice should be able to detect. In Scenario 3, he performs an *unauthorized removal* operation by removing watermarks that no one should be able to remove. Each of these three categories of attack poses its own technological challenges to the designers of secure watermarks.[2]

[2] A fourth type of attack, *unauthorized changing,* has the adversary, Bob, change the message encoded in the watermark. An example of this type of attack might occur if Alice embeds a message that says, "This Work belongs to Alice" and Bob changes it to "This Work belongs to Bob." This type of attack has received little attention in the published literature, and it is difficult to imagine a way to do it other than a combination of unauthorized removal and unauthorized embedding. That is, Bob must first remove Alice's original mark, and then embed his own mark. We therefore do not address this attack as a separate case in this chapter.

The importance of preventing each type of attack depends on the application. It is useful to analyze the security requirements of an application by specifying who is allowed to perform which operations. For a given application, we divide the world into a number of different groups and assign a set of permissions to each. The resulting operational table will determine the security properties required of the watermarking system.

Table 10.1 shows the permissions for a number of different groups in each of the three scenarios described. Note that for many entries we do not care whether a particular group can perform some operation, because it would have no direct impact on the purpose of the watermarking system. However, it may turn out to be difficult or impossible to deny a group permission for one operation while allowing them to perform another. For example, it is an open question whether a system can be designed that allows an individual to embed a watermark without giving him or her the ability to remove it. If it is impossible, we would have to deny the public the ability to embed watermarks in the copy-control application, changing the "–" in the lower left of Table 10.1 to an "N."

10.1.2 Public and Private Watermarking

Two special cases of permission combinations are shown in Table 10.2. In both cases, the world is divided into a group of *trusted* individuals, who are usually the people the watermark is designed to benefit, and the *public,* who are generally assumed to be potential adversaries. In the first case, often referred to as *private watermarking,* the public is not allowed to access the watermarks

Table 10.1 Operational table for the applications of the three scenarios presented. The label "Y" means this operation must be allowed, and the label "N" means that this operation must be prevented. The label "–" indicates that the system will work regardless of whether the operation is allowed.

	Embed	Detect	Remove
Broadcast Monitoring			
Advertiser	Y	Y	–
Broadcaster	N	N	–
Public	N	N	–
Web Reporting			
Marking Service	Y	Y	–
Reporting Service	–	Y	–
Public	N	N	N
Copy Control			
Content Provider	Y	Y	–
Public	–	Y	N

Table 10.2 Operational table for two categories of watermarking application: *private watermarking* and *public watermarking*.

	Embed	Detect	Remove
Private Watermarking			
Trusted	Y	Y	–
Public	N	N	N
Public Watermarking			
Trusted	Y	Y	–
Public	N	Y	N

in any way. In the second case, often referred to as *public watermarking,* the public is allowed to detect the watermarks, but nothing else. In this usage, "public" and "private" watermarking refer to security requirements of the application.

Similar terminology is often used to describe watermarking *algorithms*. Thus, *private watermarks* or *private watermarking systems* refer to technologies appropriate for the first set of security requirements in Table 10.2. Similarly, *public watermarks* or *public watermarking systems* refer to technologies appropriate for the second set of security requirements. Unfortunately, this usage of "public" and "private" is ambiguous, because most technologies can be used for either type of application.

As a first example, systems that employ informed detectors are often referred to as "private," on the assumption that the public does not have access to the original Works, which are needed for detection. However, there can be public watermarking applications in which the public *does* have access to the original Works. Consider a hypothetical NASA web site containing space probe imagery. Each original image is accompanied by a number of versions, which have been enhanced by some image-processing technique. Each of the enhanced images is watermarked with a text string describing the processing that has been applied. Given the original image and an enhanced image, an informed detector reveals the string describing the enhancement processes. In this scenario, we have an informed detector being used in a public watermarking application.

As a second example, systems that employ blind detection, without using keys, are often referred to as "public," on the assumption that the public knows the detection algorithm. However, if the algorithm is kept secret, it can still be used for a private watermarking application. Of course, the security of such a system would be difficult to verify before deployment (see Section 10.1.4).

As a final example, systems that employ blind detection, but *do* use keys, are equally suitable for both types of application. Here, the difference between a "public" system and a "private" system is merely a matter of how the keys are distributed.

In addition to this problem with using "public" and "private" to describe watermarking technologies, their usage invites confusion with public-key cryptography, which employs both a public and a private key. Therefore, in this book we avoid use of the terms *public watermarking* and *private watermarking*.

10.1.3 Categories of Attack

Tables 10.1 and 10.2 give a broad indication of who is permitted to perform which actions. Within the three categories of action represented by the columns there are many subtle variations. It is often possible for an adversary to perform one or more parts of an unauthorized action, even if he or she is unable to perform the action completely. For example, watermark detection generally comprises two separate steps: detecting the presence of the mark and decoding the embedded message. In some systems, it may be easy for an adversary to detect the presence of the mark but difficult to decode the message. Depending on the application, such partial attacks can be serious concerns. It is therefore necessary to differentiate each of the three broad categories of unauthorized actions into more specific *types* of attacks.

Unauthorized Embedding

The most complete type of unauthorized embedding entails the adversary composing and embedding an original message. For example, suppose Alice has a business such as that in Scenario 2, except that she charges only for embedding owner IDs and gives away the web-monitoring software for free. If Bob wants to use the web-monitoring software without paying Alice, he will try to embed his own identification mark. To do so he must compose a new message that identifies a Work as his, and build a tool for embedding it. As we shall see in Section 10.2.3, such an attack can be easily prevented with standard cryptographic techniques.

A less complete type of unauthorized embedding occurs when the adversary obtains a precomposed legitimate message, rather than composing a novel message, and embeds this message illicitly in a Work. For example, in Scenario 1, Bob wants to embed Alice's ID into his radio advertisement before broadcasting it. If Alice's watermarking system is not well designed, Bob might be able to determine the reference pattern Alice uses to encode her ID within commercials. He could then copy that pattern into his own advertisement, often without having to understand anything about how the message is encoded. This type of attack, known as a *copy attack* [248], is discussed in more detail in Sections 10.2 and 10.3.3.

Unauthorized Detection

In some applications for which the ability to detect should be restricted, we are primarily concerned with preventing people from actually decoding what

a watermark says. For example, a hospital might embed the names of patients into their X-rays. For reasons of privacy, unauthorized individuals should be prohibited from reading the marks and identifying the patients. In this case, it is sufficient to prevent the adversary from decoding the mark, and this can be done with standard cryptographic tools (see Section 10.2.3).

In other applications, however, the adversary might be satisfied with simply knowing whether or not a watermark is present, and need not know what message the watermark encodes. Strictly speaking, this is a problem of steganography, not watermarking. Nevertheless, it is conceivable that a watermarking application might arise in which the system can be compromised by an adversary who merely identifies whether or not a mark is present in a given Work. Furthermore, as we shall see in Sections 10.3.5 and 10.3.6, the ability to detect the presence of a watermark can be a great advantage to an adversary who is trying to remove marks. Thus, it is conceivable that if a watermark is to be secure against unauthorized removal it also must be secure against this type of unauthorized detection. Security against decoding, without security against detection, is not enough.

A third type of unauthorized detection, which lies between full decoding and mere detection, occurs when an adversary can distinguish between marks that encode different messages, even though he or she does not know what those messages are. That is, given two watermarked Works the adversary can reliably say whether their respective marks encode the same or different messages. This type of attack is a problem if the adversary can determine the meaning of the messages in some way other than actually decoding them. For example, in Scenario 2, Bob wants to steal Alice's customers by finding their IDs in watermarks on the Web. Alice might attempt to guard against this by using a secret code for associating watermarks with her clients, thus preventing Bob from decoding the watermarks to identify the owners of the Works. However, if Bob can reliably determine whether or not two Works contain the same mark, he can still steal Alice's business. He simply asks any prospective client to provide him with one example of a Work Alice has watermarked. When he finds other Works on the Web that contain the same watermark, he knows they belong to this customer. In this case, Bob does not need to know how Alice encodes the ID of each customer in the watermarks.

Unauthorized Removal

In general, a watermark is considered removed from a Work if it is rendered undetectable. The most extreme case of such removal would restore the Work to its original unwatermarked form. In all watermarking applications that require security against unauthorized removal, it is necessary to prevent an adversary from recovering the original. However, it is difficult to think of an example in which such prevention provides *sufficient* security.

In the vast majority of applications, the adversary will try to modify the watermarked Work such that it *resembles* the original and yet does not trigger

the watermark detector. The original Work is just one of an extremely large number of Works that fit this description. Thus, showing that an adversary cannot recover the original is only a minor step toward showing that a watermark is secure against removal.

The adversary can employ a fidelity constraint requiring that the distortion introduced by the attack has minimal perceptual impact. This is similar to the use of a fidelity constraint for embedding, but the perceptual model used by an adversary need not be the same as that used for embedding. Further, the fidelity requirements of the adversary are usually less stringent than those of embedding [73, 301]. In this case, the fidelity is measured between the watermarked Work and the attacked Work.

Within the range of Works that will satisfy the adversary, a distinction is often made between those in which the watermark is truly gone and those in which the watermark is still present but only can be detected by a more sophisticated detector than the one being used. We refer to the former as *elimination* attacks, and to the latter as *masking* attacks.

As an example of a masking attack, consider an image watermarking system in which the detector is unable to detect watermarks in images that have been rotated slightly. Thus, an adversary might "remove" a mark from an image by applying a small rotation. However, presumably a smarter detector, employing some sort of registration and search, would be able to invert the rotation and still detect the watermark.

This contrasts with an elimination attack in which the adversary tries to estimate the embedded watermark pattern and subtract it. The result of such an attack might be a fairly close approximation of the original Work (though not necessarily an exact reconstruction of the original).

Intuitively, the distinction between elimination attacks and masking attacks seems clear. When viewed in media space, however, this distinction becomes less apparent. Both the rotation attack and the estimate-and-subtract attack move points from within the detection region (i.e., watermarked Works) to points outside the detection region (i.e., unwatermarked Works). Replacing the detector with one that performs a geometric search merely changes the detection region to include the Works attacked by rotation. This is illustrated schematically in Figure 10.1. If we can counter the rotation attack by creating a more sophisticated detector, why can't we counter every attack this way?

Clearly, an attack that recovers the exact original Work cannot be countered by modifying the detector. We can identify other attacks that also cannot be addressed in this way by examining where the attacked Works lie in media space in relation to the distribution of unwatermarked content. In the case of the rotation attack, the attacked Works still have unusual properties that the embedding process gave them. That is, they are very unlikely to occur in the distribution of unwatermarked Works. Thus, if we modify the detection region to include an area around the attacked Works, we will not substantially increase the false positive probability.

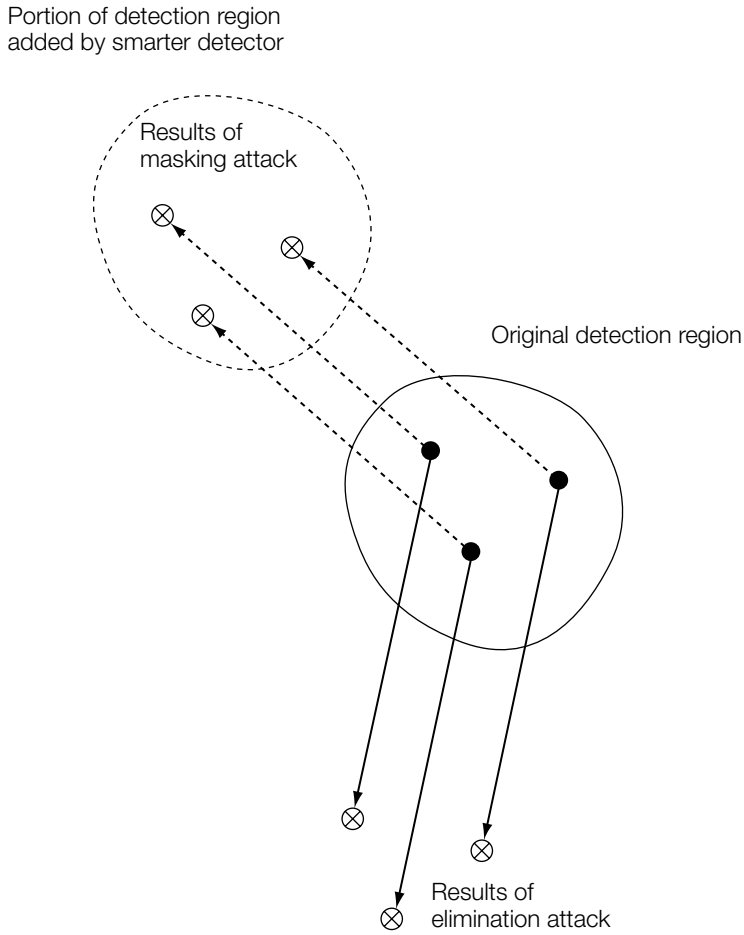


FIGURE 10.1

Illustration of elimination and masking attacks. The solid contour indicates an arbitrary detection region, defined by the original detector. The solid arrows illustrate a possible effect of an elimination attack, whereas the dashed arrows illustrate a masking attack. The masking attack can be countered by using a smarter detector, which adds the dashed contour to the detection region.

On the other hand, the attack that estimates and subtracts the reference pattern results in Works that are quite likely to arise without embedding. If we modify the detection region to include an area around these attacked Works, we will increase the false positive rate unacceptably.

Thus, we can define elimination attacks as those that move watermarked Works into regions of media space where unwatermarked Works are likely, and masking attacks as those that move watermarked Works into regions in which

unwatermarked Works are not likely. This difference between masking attacks and elimination attacks is illustrated in Figure 10.2.

The distinction between elimination attacks and masking attacks can be important in applications in which it is feasible to modify the detector after the attack is discovered. For example, in a transaction-tracking application, only the owner of a Work might need to detect watermarks. If an adversary attempts to thwart the system by applying a masking attack, so that the owner's watermark

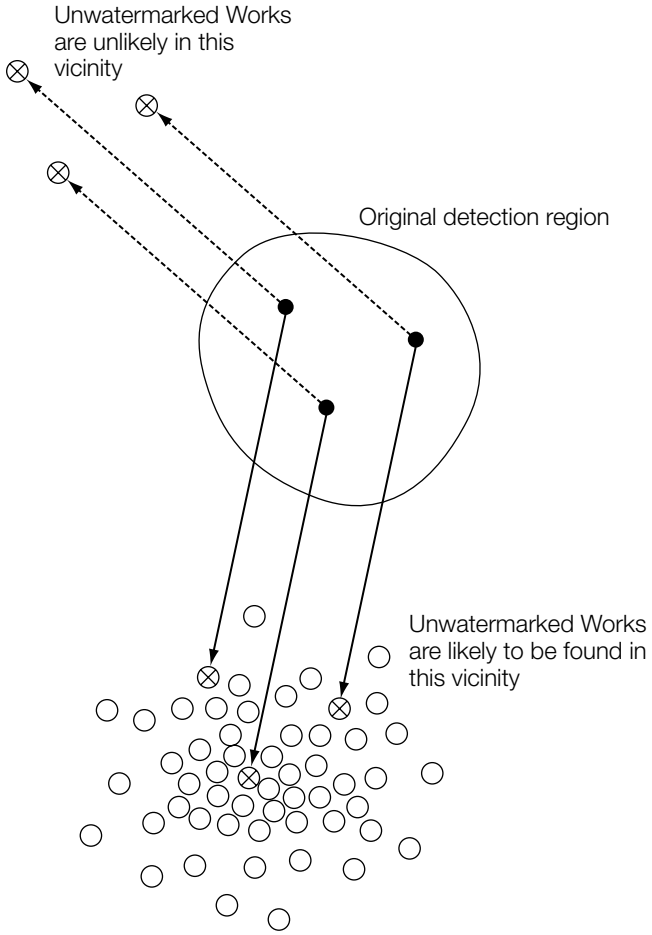


FIGURE 10.2

Illustration of the significant difference between elimination and masking attacks. The crowd of open circles indicates a region of media space in which unwatermarked Works are very likely. The elimination attack (solid arrows) moves watermarked Works into this region. The masking attack (dashed arrows) moves watermarked Works into a region in which unwatermarked Works are not likely.

detector will not detect the marks in pirated Works, the owner has a chance of upgrading his or her detector to a smarter one that can detect the distorted watermark. On the other hand, if the adversary can apply an elimination attack, the owner has little chance of detecting the marks, even with a better detector. Thus, in applications in which it is possible to update the detector, masking attacks are less serious than elimination attacks.

System Attacks

Before concluding this discussion of types of attacks on watermarking systems, we must point out that not all attacks are attacks on the watermarks themselves. An adversary can often thwart the system without having to perform any of the previously discussed unauthorized actions. Attacks that exploit weaknesses in how the watermarks are used, rather than weaknesses in the watermarks themselves, can be broadly referred to as *system attacks*.

As a simple example of a system attack, consider a copy-control application in which every recording device includes a computer chip for detecting the watermark. An adversary might open the recorder and just remove the chip, thereby causing the recorder to allow illegal copying. In such an attack, the security of the watermark itself is irrelevant.

Of course, in this book we are discussing only watermarking technology, and therefore system attacks such as removal of a detector chip are outside our scope. However, it is important to keep these types of attacks in mind when designing a system that employs watermarks. The adversary will always attack the weakest link in the security chain.

10.1.4 Assumptions about the Adversary

In evaluating the ability of watermarking technologies to meet the types of requirements laid out in Tables 10.1 and 10.2, it is necessary to make some assumptions about the capabilities of the adversary. What does he or she know about the watermarking algorithm? What tools does he or she have at his or her disposal? For example, if he or she is trying to remove a watermark, does he or she know how the mark was embedded? Does he or she have a detector he or she can experiment with?

If the Attacker Knows Nothing

The simplest assumption is that the adversary knows nothing about the algorithms and has no special tools (such as a watermark detector). Under these circumstances, the adversary must rely on general knowledge of the weaknesses from which most watermarking algorithms suffer. For example, suppose an adversary thinks a Work might be watermarked and wants to remove that mark. He or she might try applying various distortions he or she knows mask most watermarks, such as noise-reduction filters, extreme compression, or small geometric or temporal distortions. This is the basic approach taken

by the Stirmark program [247], which often succeeds in making watermarks undetectable.

If the Attacker Has More Than One Watermarked Work

In some cases, it is possible for an adversary to obtain multiple watermarked Works. The adversary can often exploit this situation to remove watermarks, even when he or she does not know the algorithm. Attacks that rely on possession of several watermarked Works are known as *collusion attacks*.

There are two basic types of collusion attacks. In one, the adversary obtains several different Works that contain the same watermark and studies them to learn how the algorithm operates. The simplest example of such an attack has the adversary averaging several different marked Works. If all Works have the same reference pattern added to them, this averaging would yield something close to that pattern. The averaged pattern then can be subtracted from Works to remove the watermark.

A variation on this first type of collusion attack can be performed when watermarks are embedded redundantly in tiles (see Chapter 9). If the same watermark pattern is added to several pieces of a Work, the adversary can view those pieces as separate Works and perform a collusion attack to identify the pattern. This type of attack was successfully applied to one of the audio watermarking systems proposed for use in the Secure Digital Music Initiative (SDMI) [43].

The other type of collusion attack involves the adversary obtaining several copies of the same Work, with different watermarks in them. In this case, the adversary might be able to obtain a close approximation of the original Work by combining the separate copies. The simplest combination is to average the copies, thereby mixing the different marks together and reducing their amplitude. More sophisticated techniques, which can remove some watermarks using a small number of copies of a Work, are described in Stone [389].

Boneh and Shaw [45] have posed the problem of designing *collusion-secure codes* (i.e., codes that are secure against this second type of collusion). Suppose a Work is distributed to n individuals, with a distinct code word embedded in each copy. The code is said to be *c-secure* if, when up to c individuals collude, there is a high probability that the resulting Work will contain enough information to identify at least one of the colluders.

This can occur if portions of the code words for all colluders are identical. When comparing their respective copies, the colluders learn nothing about how to corrupt these portions of the watermarks, and therefore we can assume that they will be unaffected by the attack. If these unaffected portions of the code words carry enough information, we can identify at least one of the colluders. Following Boneh and Shaw's lead, several researchers have studied the design of binary c-secure codes [266, 270, 307, 401], and alternative formulations have been developed [410].

Recently, Cayre *et al.* [66, 67] have formalized the case where the attacker has copies of watermarked Works. Specifically, they consider the cases of:

1. the Watermarked Only Attack in which the adversary has access to a number of watermarked Works,
2. the Known Message Attack in which the adversary has access to a number of watermarked Works and their associated messages, and
3. the Known Original Attack in which the adversary has access to a number of watermarked Works and their associated original Works. In this case, the adversary's goal is to acquire information about the secret key (see Section 10.2) in order to attack other watermarked Works for which the originals are unavailable.

If the Attacker Knows the Algorithms

For systems that require a very high level of security, the assumption that the adversary knows nothing about the watermarking algorithm is generally considered unsafe. It is often difficult to keep the algorithms completely secret. Furthermore, if an algorithm is to be kept secret, only a small number of researchers can be allowed to study it. This means that serious security flaws can go undiscovered until after the system is deployed.

For these reasons, the cryptographic community advocates assuming that the adversary knows everything about the algorithm except one or more secret keys. This is known as *Kerckhoffs' assumption* [231]. Not only do cryptographers assume the adversary knows the algorithm, they often ensure that he or she does by publishing it themselves. This allows their peers to study the algorithm and try to find flaws in it.

An adversary who has complete knowledge of a watermarking algorithm may be able to find and exploit weaknesses in the detection strategy. For example, he or she may be able to identify specific distortions for which the detector cannot compensate, and thus apply a successful masking attack. In addition, knowledge of the watermarking algorithm may help an adversary determine the secrets that describe a specific watermark. For example, consider an algorithm that creates a tiled watermark pattern (like our block-based algorithms). Knowing that the extraction process averages tiles, the adversary might be able to determine tile size and create an estimate of the reference mark. Once the adversary has gained the secret, he might be able to perform unauthorized embedding, unauthorized detection, or unauthorized removal of the watermark.

If the Attacker Has a Detector

In all of the previously discussed cases, we have assumed that despite what the adversary knows about the watermarking algorithm he or she does not have any special tools at his or her disposal. If, however, the application dictates that the adversary must have permission to perform some action, we must assume that

he or she has the tools required for that action. The case that has received the most study is the one in which the adversary is allowed to detect watermarks, but not allowed to remove them. In this case, we must assume that the adversary has a watermark detector.

Even if the adversary knows nothing about the algorithm, access to a detector gives him or her a tremendous advantage in attacking the watermark. We can think of the detector as a "black box" or "oracle" to which the adversary can submit a modified Work and find out whether or not it is inside the detection region. By making iterative changes to the Work, and testing after each change, it is possible to learn a great deal about how the detection algorithm operates. For example, this approach is exploited in the *sensitivity analysis* attack, which is described in detail in Section 10.3.5.

An even more difficult security problem arises if the adversary has a detector and knows how it works. In essence, he or she knows the detection algorithm and any key required to detect the specific marks in question. To date, no watermarking algorithm has been proposed that can prevent unauthorized removal under these conditions. However, as discussed in the next section, a few researchers are attempting to apply the principles of asymmetric-key cryptography to the problem.

10.2 WATERMARK SECURITY AND CRYPTOGRAPHY

In this section we discuss the relationships between encryption and watermarking. We examine cryptographic techniques for restricting watermark embedding and detection. In many cases, the problems of unauthorized embedding and detection are directly analogous to problems studied in cryptography and can be solved by straightforward application of cryptographic tools. The problem of unauthorized removal, however, does not have a direct analog in the world of cryptography, and therefore it is less obvious how cryptographic techniques might be used to solve it. A discussion of this topic concludes the section. (Those unfamiliar with cryptographic concepts such as one-way hashing and asymmetric keys may wish to read Section A.3 of Appendix A before proceeding.)

10.2.1 The Analogy between Watermarking and Cryptography

Watermark embedding and detection can sometimes be considered analogous to encryption and decryption. In symmetric-key cryptography, we have an encryption function, $E_K(\cdot)$, that takes a key, K, and some *cleartext,* m, and produces a *ciphertext,* m_c. That is,

$$m_c = E_K(m). \tag{10.1}$$

The ciphertext can be decrypted to reveal the original plaintext message using a decryption function, $D_K(\cdot)$, and the corresponding key, K. That is,

$$m = D_K(m_c). \tag{10.2}$$

In watermarking, we have an embedding function, $\mathcal{E}(\cdot)$, that takes a message, m, and an original Work, $\mathbf{c_o}$, and outputs a watermarked Work, $\mathbf{c_w}$. Similarly, we have a detection function, $\mathcal{D}(\cdot)$, that takes a (possibly) watermarked Work and outputs a message. In many watermarking systems, the mapping between watermarked Works and messages is controlled by a watermark key, K. In the case of informed detection, the detection key can be considered to include a function of the original Work. That is, we can think of a system with informed detection as simply associating a unique key with each Work. Thus, we can characterize almost any watermarking system by the equations

$$\mathbf{c_w} = \mathcal{E}_K(\mathbf{c_o}, m) \tag{10.3}$$

and

$$m = \mathcal{D}_K(\mathbf{c_w}), \tag{10.4}$$

which are similar to Equations 10.1 and 10.2.

With this analogy between cryptography and watermarking, it is reasonable to expect that the problems outlined in Section 10.1.3 might be resolved using appropriate cryptographic methods. In Sections 10.2.2 and 10.2.3, we examine the use of cryptographic tools for restricting watermark detection and embedding, respectively. On the other hand, as we discuss in Section 10.2.4, the problem of unauthorized watermark removal does not have a direct analog in cryptography. We discuss the limitations of cryptographic tools in preventing unauthorized watermark removal. The reader is directed to [98] for further discussion of cryptography and watermarking.

10.2.2 **Preventing Unauthorized Detection**

Let us first examine the problem of confidential communication between Alice and Bob. In the context of watermarking, this requires that they prevent the unauthorized detection and decoding of the watermark message. Ideally, they would like to use a watermarking system that is secure against these categories of attack.

Unfortunately, it is sometimes impractical to make a watermarking system that is truly secure against unauthorized detection and decoding of messages. In general, for a watermarking system to be secure against these attacks, it must have a very large keyspace. That is, there must be a very large set of distinct keys from which Alice and Bob's key is drawn. If the keyspace is too small, an adversary (Eve) can identify the right key by brute-force search (assuming she knows the watermarking algorithm, which is the safest assumption). The

difficulty is that the design of a watermarking algorithm is often a compromise between many conflicting requirements, and may be constrained to have a small keyspace.

In such circumstances, the problem of unauthorized decoding can be solved by a straightforward application of cryptographic algorithms. We add a level of traditional encryption to the system. The message is encrypted before it is embedded, and decrypted after it is detected. Such a system requires two keys: the watermark key, K_w, controls the watermarking layer, and the encryption key, K_c, controls the encryption layer. Thus, at the embedder, the watermarked work is given by

$$\mathbf{c_w} = \mathcal{E}_{K_w}(\mathbf{c_o}, m_c) = \mathcal{E}_{K_w}(\mathbf{c_o}, E_{K_c}(m)). \tag{10.5}$$

At the detector, the reverse process is employed:

$$m = D_{K_c}(\mathcal{D}_{K_w}(\mathbf{c_w})). \tag{10.6}$$

Such a system is illustrated in Figure 10.3. When the watermark uses sequences of symbols to encode messages (see Chapter 4), the division of labor between the encryption and watermarking layers of this system can be nicely described: the encryption layer obscures messages, whereas the watermarking layer obscures symbols.

As indicated in Figure 10.3, the encryption and watermarking layers can be viewed as analogous to two layers of a networking system. Networking systems are generally divided into several layers, each of which is responsible for a specific task. The two layers shown here are the *message layer,* responsible for determining

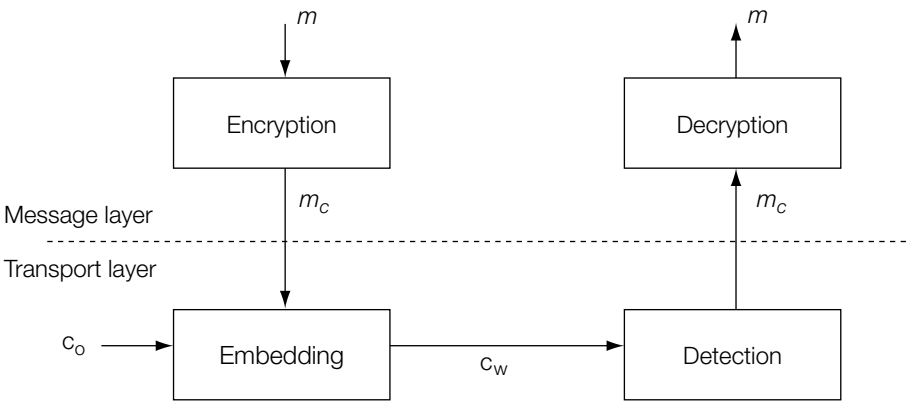


FIGURE 10.3

A two-layered watermarking system. The top layer is responsible for ensuring that messages are secure. The bottom layer is responsible for ensuring that they are securely transported.

what messages to transmit over the network, and the *transport layer,* responsible for ensuring that transmitted messages arrive uncorrupted. In the system of Figure 10.3, encryption is part of the message layer, and the watermarking system, together with the cover Work, is part of the transport layer.

The addition of an encryption layer ensures message security (i.e., that someone who detects the watermark cannot decode its meaning). In addition, in certain systems it can prevent the adversary from determining the presence of a watermark. Specifically, if a watermark system is designed to determine the presence of a watermark by distinguishing between valid and invalid messages (see Section 4.3.1), it may be impossible to identify valid messages without decrypting. Hence, an adversary that has the watermark key but does not have the cipher key would be unable to determine the presence of a watermark [21, 100].

Unfortunately, in most watermarking systems encryption does not prevent the detection of the encrypted message. This is true for systems that determine the presence of a message by comparing a detection statistic against a threshold (see Sections 4.3.2 and 4.3.3). This is also true for systems in which the presence of watermarks can be detected by other means. For example, a watermark embedded in the least significant bits of a Work can produce telltale statistical artifacts in the Work's histogram. These artifacts can be exploited to distinguish between watermarked and unwatermarked Works, even if the marks themselves are indistinguishable from noise [153, 444].

This limitation of encryption is intuitively clear from the message layer/transport layer analogy of Figure 10.3. Preventing unauthorized detection without decoding is a problem of transport security, in that the adversary is essentially detecting the fact that a message is being transmitted. In contrast, encryption is part of the message layer and is concerned with issues of message confidentiality, authentication, integrity, and nonrepudiation [362]. Thus, unauthorized detection is unlikely to be prevented by any straightforward application of cryptographic tools.

10.2.3 Preventing Unauthorized Embedding

Now let's consider the problem of unauthorized embedding. This problem is closely analogous to the cryptographic problem of authenticating the sender, which can be solved with either asymmetric-key cryptography or cryptographic signatures, depending on the size of the messages. Thus, a straightforward approach to preventing unauthorized watermark embedding would be to use either of these cryptographic tools in the message layer of Figure 10.3. That is, before embedding a message, we could either encrypt it using asymmetric-key cryptography or append a cryptographic signature.

These straightforward approaches prevent one type of unauthorized embedding attack, discussed in Section 10.1.3; namely, an attack in which the adversary composes a novel message and embeds it. Even if the adversary

knows the embedding algorithm and watermark key, K_w, he or she cannot correctly encrypt his or her message, or create a valid cryptographic signature, unless he or she also knows the encryption key. Thus, as long as Bob keeps his private encryption key secret, Alice can verify that Bob is the source of an embedded message by using the corresponding public decryption key.

However, these simple solutions do not address the second type of unauthorized embedding, discussed in Section 10.1.3; namely, an attack in which the adversary finds an existing valid message and embeds it in a Work. The copy attacks discussed in Section 10.3.3 are examples in which the adversary extracts the valid watermark from a legitimately watermarked Work and copies it to an unwatermarked target Work. If the watermark is encrypted with a private key, it is copied over in its encrypted form and is still decodable with the corresponding public key. If the watermark has a cryptographic signature, the signature is copied along with it and still matches the hash of the message. Thus, the forgery will appear completely legitimate.

One interpretation is that the message embedded in the Work is indeed a valid message but the cover Work is wrong. Because watermarks by definition carry information about the Works in which they are embedded, the messages cannot be properly interpreted without reference to the Work. The message "You may copy this" embedded in an image of Winnie the Pooh means that you are authorized to copy that particular image and no other. Thus, strictly speaking, the cover Work must be considered an implicit part of the message. To guard against copy attacks, then, Alice must be able to validate the *entire* watermark message, including its link to the cover Work.

How can this link be validated? One possibility is to append a description of the cover Work into the embedded message, which is then protected with either asymmetric-key encryption or a cryptographic signature. When Alice receives the watermarked Work, she validates this description and compares it with the Work.

There are several variations on the basic idea of appending a description of the cover Work to an embedded message. One obvious approach would be to append the entire cover Work to the message before computing a cryptographic signature. Then the message and the signature would be embedded. Unfortunately, this approach would fail because the embedding process modifies the Work, which makes the signature invalid.

To work around this problem, we might sign only a portion of the Work, such as the lowest-frequency components, rather than the entire Work. If the embedding process is designed in such a way that it does not change this portion of the Work, the signature should still be valid after the watermark is embedded. Thus, to embed a watermark, Bob performs the following steps:

1. Construct a description of the cover Work based on information unlikely to change, such as the lowest-frequency components.

2. Compute a one-way hash of the watermark message concatenated with the description of the cover Work.
3. Encrypt the hash with a private key, thus obtaining a cryptographic signature.
4. Embed the watermark message along with the signature, using an embedding algorithm that does not change the description computed in step 1.

To detect and validate a watermark, Alice performs the following steps:

1. Detect and decode the watermark, obtaining a message and a cryptographic signature.
2. Construct the same description of the cover Work that Bob constructed.
3. Compute the one-way hash of the watermark message concatenated with the description of the cover Work.
4. Decode the cryptographic signature using Bob's public key.
5. Compare the decoded signature against the hash of the message and cover Work description. If they are identical, Alice knows that Bob composed this message and embedded it in this Work.

These procedures are illustrated in Figure 10.4.

One difficulty with this system is that if the Work is expected to suffer some degradation between embedding and detection, the description used in the watermark signature must be extremely robust. If even 1 bit of this description changes, the one-way hash will change completely, and the signature will be invalid. To alleviate this problem, we might use a different approach, illustrated in Figure 10.5. Here, Bob still computes a description of the Work and uses it in a cryptographic signature. However, when embedding the message and the signature, he also embeds the description. Alice, then, can obtain the exact description of the Work Bob used to compute the signature. She uses the signature to verify that all of the embedded information (i.e., the message and the description) is valid. She then performs an inexact comparison between the embedded description and a description of the received Work. For example, she might calculate the correlation between these two descriptions and compare it against a threshold. This system allows the description that Alice computes to differ slightly from the one Bob computed, without invalidating the signature.

In either of these systems, the method of describing the Work must be designed to describe the Work's most perceptually significant features. More precisely, it must be extremely difficult to find two perceptually different Works that yield the same description. Otherwise, when copying a watermark from a legitimate Work to some target Work, the adversary might be able to modify

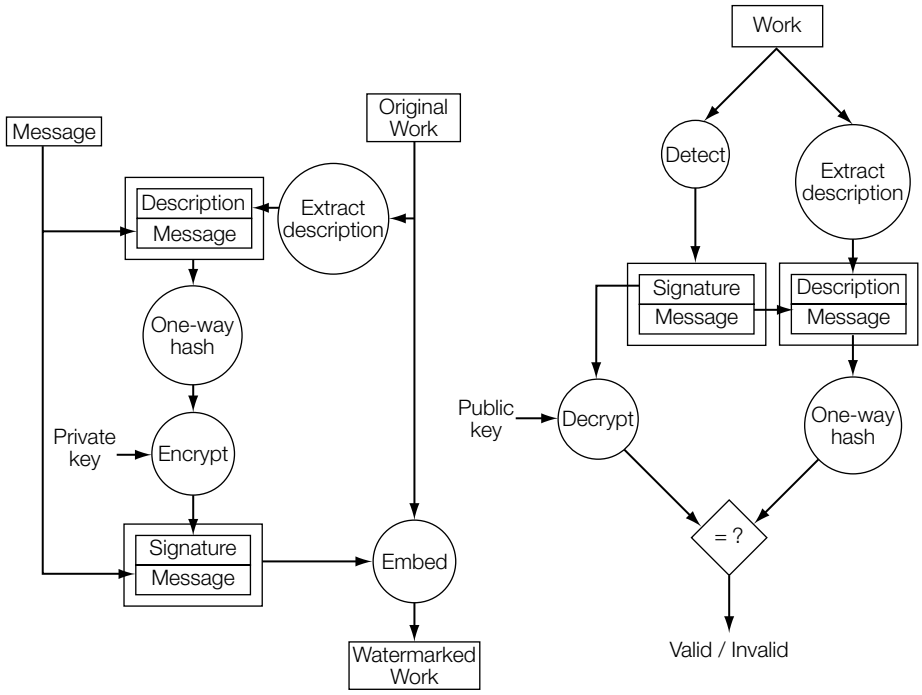


FIGURE 10.4

A method of linking watermarks to their cover Works. During embedding (a), a one-way hash is applied to both the watermark message and a robust description of the Work. This is then encrypted with a private key to form a cryptographic signature, which is embedded along with the message. For watermark validation (b), the message and signature are first extracted from the Work. The message is then sent through a one-way hash along with a robust description of the Work, calculated in the same way as during embedding. If the result matches the decrypted signature, the Work and watermark are valid.

the target Work so that its description matches that of the legitimate Work, and thus make the signature valid.

The previously described systems for linking a watermark to its cover Work illustrate an important limitation of the network layer analogy illustrated in Figure 10.3. The power of this layered communication model is that each layer can be considered independent of the other layers, provided an interface between the levels is clearly defined. However, for watermarking, these systems imply that if we are to link the message with the Work, the message (cryptographic) layer must be tailored to the specific transport (watermarking) system being used. First, it must incorporate knowledge of the cover Work into the cryptographic protection of the message. Second, it must know the watermarking

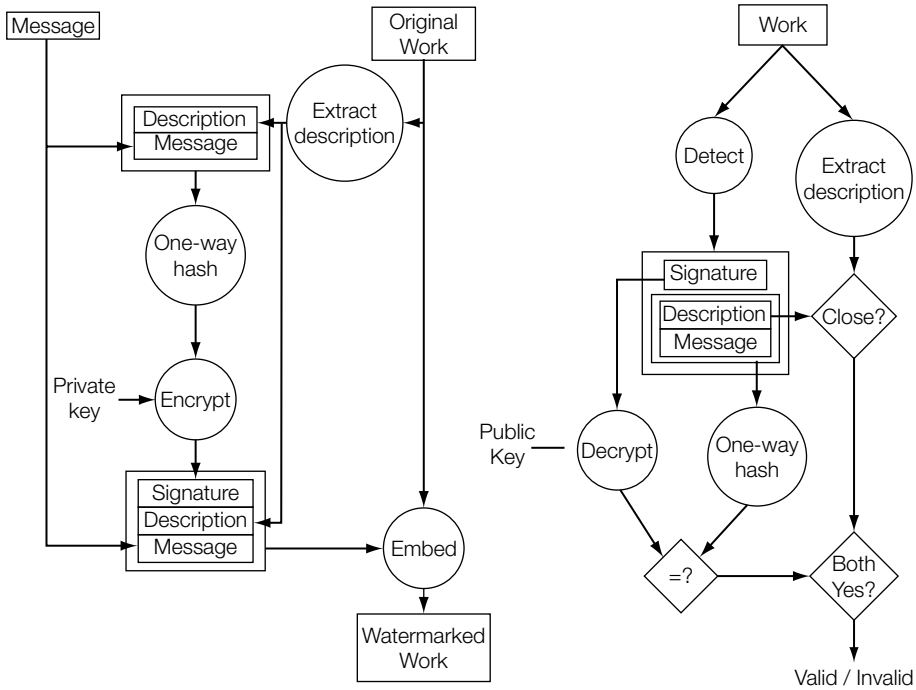


FIGURE 10.5

An alternative method for linking watermarks to cover Works. The embedding process (a) is very similar to that shown in Figure 10.4, except that the robust description of the Work is embedded along with the message and signature. In validation (b), the signature is used to validate the message and the embedded description. The embedded description is then compared against a newly computed description to verify that the Work matches the one in which the watermark was originally embedded. This match need not be exact for the watermark to be validated.

algorithm to determine what properties of the Work are unaffected by the embedding process.

10.2.4 **Preventing Unauthorized Removal**

Although the analogy between watermarking and cryptography is useful to counter unauthorized embedding and decoding, its utility is not clear in the case of unauthorized removal. This is because there is no direct analogy between unauthorized removal and any of the main cryptographic problems [98]. In the terminology of Figure 10.3, unauthorized removal is strictly a problem with the transport layer: an adversary prevents a transmitted message from arriving.

When the adversary is neither authorized to embed nor to decode, we turn to spread spectrum techniques, rather than cryptographic techniques, to prevent unauthorized removal. The problem becomes similar to that of secure military communications, for which spread spectrum is used to ensure delivery of the message.

As discussed in Section 3.2.3, spread spectrum communication transmits a signal over a much larger spectrum than the minimum needed. The signal spreading is determined by a key that a receiver also must be in possession of in order to detect the signal. Spread spectrum communications are very difficult to jam or remove and the probability of unauthorized detection is extremely small.

When spread spectrum is used for secure military communications, the spreading function is designed to make the modulated message appear to an adversary like background noise. This makes it very difficult for an adversary to determine if a transmission is occurring. When spread spectrum is used for secure watermarking, the added patterns also can be designed to have the same spectral characteristics as the background, which in this case is the cover Work. Again, this makes it difficult for an adversary to determine whether a watermark is present in a Work and helps maintain the fidelity of the cover Work [337]. Interestingly, in Su and Girod [392] it is concluded that the form of watermark most secure against Wiener filtering attacks, discussed in Section 10.3.2, is one whose power spectrum resembles that of the cover Work.

Because spread spectrum communication spreads the signal energy over a very large spectrum, and an adversary is unable to transmit sufficient power over such a broad spectrum, jamming is not practical. Similarly, for watermarking, an adversary is unable to add sufficient noise to the cover Work to be confident of eliminating the watermark without seriously degrading the fidelity.

Spread spectrum techniques are useful when the adversary is not authorized to embed or decode. Unfortunately, there are many applications, including copy control, for which the adversary must be assumed to have the ability to detect marks. In this case, spread spectrum techniques cannot ensure secure transmission. Many researchers suspect that such applications are inherently insecure. However, this conjecture has not been proven, and there are a number of dissenting points of view.

One line of research being investigated is based on the belief that watermarking can be made secure by creating something analogous to an asymmmetric-key cryptographic system. That is, we would like the watermark embedder to use one watermark key, and the watermark detector to use a different watermark key. The assumption is that knowledge of the detection key is not sufficient to allow an adversary to remove a watermark. A few such asymmetric-key watermarking systems have been proposed, which either use different keys at the embedding and detection stages or a key only for embedding and no key for detection [128, 157, 158, 422]. Although these proposals possess an asymmetry analagous to asymmetric-key cryptography, they have not yet been shown

to prevent watermark removal. In fact, most of these are probably susceptible to attacks such as sensitivity analysis (Section 10.3.5).

The aim of these asymmetric-key watermarking systems is similar to that of asymmetric-key encryption systems. Recall that in asymmetric-key cryptography we have two different descriptions of a single mapping between strings of cleartext and strings of ciphertext, based on two different keys. Similarly, in asymmetric-key watermarking, the aim is to construct two descriptions of a mapping between Works and embedded messages. As in asymmetric-key cryptography, these two descriptions should make different operations computationally feasible. One, based on the embedding key, should make it easy to map from messages to Works. The other, based on the detection key, should make it easy to map from Works to messages.

However, there are some fundamental differences between watermarking and cryptography that make the standard asymmetric-key encryption systems unsuitable for our application. First, in watermarking, the mapping between Works and messages must be many-to-one, so that a given message may be embedded in any given Work. In asymmetric-key cryptography, the mapping between cleartext and ciphertext is always one-to-one. Second, the set of Works that all map into the same message—the detection region for that message—must be clustered in such a way as to produce robust watermarks. That is, if a watermarked Work that maps into a given message is modified slightly, the resulting Work should still map into the same message. Asymmetric-key encryption algorithms map cleartext into ciphertext in a pseudo-random fashion. A small change in the cleartext results in a large change in the ciphertext, and vice versa. These differences are illustrated schematically in Figure 10.6.

Asymmetric-key watermarking can also be viewed geometrically as a matter of constructing two different descriptions of a single shape—the detection region for a given message—rather than as two different descriptions of a mapping. Under this view, the description based on the embedding key makes it easy to find a point within the shape (a watermarked Work) that is close to a point outside the shape (an unwatermarked original). The detection key makes it easy to determine whether a point is within the shape, but makes it difficult to find a nearby point outside the shape. It is an open question whether such a pair of descriptions of a shape can be constructed.

It should be noted that asymmetric-key watermarking might not be the only way to prevent unauthorized removal by adversaries who know the detection algorithms and keys. For example, it might be possible to make a secure system by making the detection region different from the embedding region. The embedding region would be simple, so that it would be easy to find a point within it that is near a given point outside. The detection region would be complicated, so that no matter how it is described it would never be easy to find a point outside that is close to a given point inside. If the detection region completely contained the embedding region, embedding and detecting watermarks would be easy, but it would be difficult to remove them, *even*

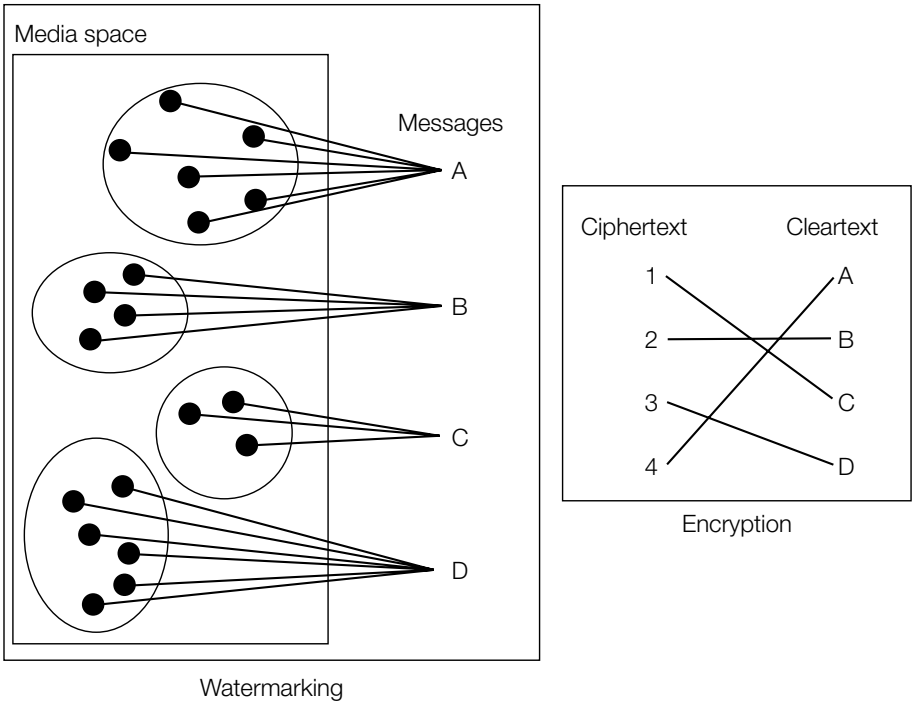


FIGURE 10.6

Illustration of the differences between watermark mappings and cryptographic mappings. Watermark mappings are many-to-one and must map groups of similar Works into the same message (i.e., the detection regions for different messages must be robust). Cryptographic mappings are one-to-one and random.

with complete knowledge of both the embedding and detecting keys. If such a system can be constructed, it would provide security against removal without the need to keep keys secure.

10.3 SOME SIGNIFICANT KNOWN ATTACKS

In this section we describe a number of known attacks on the security of watermarking systems. Some, such as copy attacks and ambiguity attacks, can be successfully addressed, either directly or indirectly, with the cryptographic tools previously described. For others, such as sensitivity analysis attacks and gradient descent attacks, reliable countermeasures are not known. We can, however, identify the basic security weaknesses that allow these attacks to be successful and thus suggest the direction toward more secure watermarking methods.

We start this section with an attack that cannot be addressed within the confines of watermarking. The scrambling attack, rather, is an attack on the *system* in which watermarking is being used. As such, it must be addressed at the system level and, as we will see, this often may be impossible.

10.3.1 Scrambling Attacks

A *scrambling attack* is a system-level attack in which the samples of a Work are scrambled prior to presentation to a watermark detector and then subsequently descrambled. The type of scrambling can be a simple sample permutation or a more sophisticated pseudo-random scrambling of the sample values. The degree of scrambling necessary depends on the detection strategy. For example, a permutation of 8×8 blocks of an image will not prevent the D_BLK_CC detector from finding the watermark, but will thwart the D_LC detector.

The only constraint is that the scrambling be invertible or near invertible. A near-invertible or lossy scrambling attack should result in a Work that is perceptually similar to the watermarked Work.

A well-known scrambling attack is the *mosaic attack,* in which an image is broken into many small rectangular patches, each too small for reliable watermark detection [329]. These image segments are then displayed in a table such that the segment edges are adjacent. The resulting table of small images is perceptually identical to the image prior to subdivision. This technique can be used in a web application to evade a web-crawling detector. The scrambling is simply the subdivision of the image into subimages, and the descrambling is accomplished by the web browser itself.

The mosaic attack is convenient for an adversary because most web browsers will correctly "descramble" the image. More general scrambling attacks require the receiver of the pirated Works to obtain a descrambling device or program. For example, to circumvent proposed copy-control systems in video recorders, a consumer might purchase both an inexpensive scrambler and descrambler. By scrambling the input to the video recorder, the adversary can cause the watermark to be undetectable, so that the recorder would allow recording. On playback, the recorded, scrambled video is passed through the descrambling device prior to viewing. Although legislation in many countries forbids the sale of devices solely intended to circumvent copyright law, an argument could be made that the scrambling devices also have legitimate uses, such as allowing adults to prevent their children from viewing inappropriate content.

10.3.2 Pathological Distortions

For a watermark to be secure against unauthorized removal, it must be robust to any process that maintains the fidelity of the Work. This process may be a *normal* process, in which case we are requiring that a secure watermark

be robust. However, it may also be a process unlikely to occur during the normal processing of the Work. Any process that maintains the fidelity of the Work could be used by an adversary to circumvent the detector by masking or eliminating the watermark. The two most common categories of such pathological distortions, geometric/temporal distortions (attacks on synchronization) and noise removal distortions, are described in the following.

Synchronization Attacks

Many watermarking techniques, including all systems presented in this book, are sensitive to synchronization. By disturbing this synchronization, an adversary attempts to mask the watermark signal. Examples of simple synchronization distortions include delay and time scaling for audio and video, and rotation, scaling, and translation for images and video. These simple distortions can be implemented such that they vary over time or space. More complex distortions include pitch-preserving scaling and sample removal in audio, and shearing, horizontal reflection, and column or line removal in images. Even more complex distortions are possible, such as nonlinear warping of images. A number of these distortions are applied by the StirMark [247] program, a tool that applies many different distortions to a watermarked Work.

Linear Filtering and Noise Removal Attacks

Linear filtering also can be used by an adversary in an attempt to remove a watermark. For example, a watermark with significant energy in the high frequencies might be degraded by the application of a low-pass filter. In addition, any watermarking system for which the added pattern is "noiselike" is susceptible to noise-removal techniques.

For certain types of watermarking systems, Su and Girod [392] show that Wiener filtering is an optimal linear-filtering/noise-removal attack. It is argued that Wiener filtering is a worst-case linear, shift-invariant process when (1) the added pattern is independent of the cover Work, (2) both the Work and the watermark are drawn from zero-mean Normal distributions, and (3) linear correlation is used as the detection statistic.

The authors show that the security of a watermark against Wiener filtering can be maximized by selecting the power spectrum of the added pattern to be a scaled version of the power spectrum of the original Work, as

$$|\mathbf{W_a}|^2 = \frac{\sigma^2_{\mathbf{w_a}}}{\sigma^2_{\mathbf{c_o}}}|\mathbf{C_o}|^2, \tag{10.7}$$

where $|\mathbf{C_o}|^2$ is the power spectrum of the cover Work, $|\mathbf{W_a}|^2$ is the power spectrum of the added pattern, and $\sigma^2_{\mathbf{w_a}}$ and $\sigma^2_{\mathbf{c_o}}$ are the variances of the distributions

from which the added pattern and cover Work are drawn. This is an intuitive result suggesting that when the watermark "looks like" the cover Work an adversary will have a difficult time separating the two.

10.3.3 **Copy Attacks**

A *copy attack* occurs when an adversary copies a watermark from one Work to another. As such, it is a form of unauthorized embedding.

The term *copy attack* was introduced in Kutter *et al.* [248], which describes a particular method of implementing it. Given a legitimately watermarked Work, $\mathbf{c_{1w}}$, and an unwatermarked target Work, $\mathbf{c_2}$, this method begins by applying a watermark removal attack to $\mathbf{c_{1w}}$ to obtain an approximation of the original, $\hat{\mathbf{c}}_1$. For this step, the authors proposed using a nonlinear noise-reduction filter. However, in principle, any method of estimating the original Work should suffice. The next step is to estimate the added watermark pattern by subtracting the estimated original from the watermarked Work:

$$\hat{\mathbf{w}}_\mathbf{a} = \mathbf{c_{1w}} - \hat{\mathbf{c}}_1. \tag{10.8}$$

Finally, the estimated watermark pattern is added to the unwatermarked Work to obtain a watermarked version, thus:

$$\mathbf{c_{2w}} = \mathbf{c_2} + \hat{\mathbf{w}}_\mathbf{a}. \tag{10.9}$$

This method has been shown to work with several commercial image watermarking systems, and it would clearly work with our E_BLIND/D_LC and E_BLK_BLIND/D_BLK_CC example systems.

The method of Kutter *et al.* [248] depends on being able to obtain a reasonable approximation of the original Work, $\hat{\mathbf{c}}_1$. It might therefore seem that the attack can be countered by ensuring that it is infeasible to obtain such an approximation. However, depending on the watermarking algorithm, there can be other ways of performing a copy attack that do not involve reconstructing the original. As a simple example, consider a watermark embedded in the least significant bits (LSBs) of a Work, $\mathbf{c_{1w}}$. It is quite infeasible to reconstruct the original version of this Work, because the LSBs of the original were probably random. However, it is trivial to perform a copy attack: we need only copy all LSBs of $\mathbf{c_{1w}}$ into the LSBs of the target Work, $\mathbf{c_2}$. Thus, the fact that reconstructing the original is virtually impossible with the LSB watermarking algorithm does not provide sufficient protection against a copy attack. A more sophisticated example, discussed in Section 11.3.1, can be found in Holliman and Memon [195].

A possible approach to countering copy attacks is to use cryptographic signatures that link the watermark to the cover Work, as suggested in Section 10.2.3. These ensure that if a watermark is successfully copied from a legitimately watermarked Work to an unwatermarked Work, the detector

will be able to determine that the watermark does not belong with the latter Work.

10.3.4 Ambiguity Attacks

Ambiguity attacks[3] create the appearance that a watermark has been embedded in a Work when in fact no such embedding has taken place. An adversary can use this attack to claim ownership of a distributed Work. He or she may even be able to make an ownership claim on the original Work. As such, ambiguity attacks can be considered a form of unauthorized embedding. However, they are usually considered system attacks.

Ambiguity Attacks with Informed Detection

The ambiguity attack described in Craver *et al.* [102] works against systems that use an informed detector. The adversary, Bob, defines his fake watermark to be a randomly generated reference pattern. He then subtracts this pattern from the watermarked Work that Alice has distributed, to create his fake original. Although the fake watermark has a very low correlation with Alice's distributed Work, it has a high correlation with the difference between the distributed Work and the fake original. If we assume that the reference pattern selected by Bob is uncorrelated with the one used by Alice (this will be the case with very high likelihood), the difference between Alice's true original and Bob's fake original will also have high correlation with the fake watermark. Thus, a situation is created in which both Alice and Bob can make equal claims of ownership. Although Alice can claim that both the distributed Work and Bob's "original" are descendants of her original, in that her watermark is detected in both, Bob can make the same claim. Bob's watermark is detected in both the distributed Work and Alice's true original.

Ambiguity Attacks with Blind Detection

An ambiguity attack against a system that uses blind detection can be performed by constructing a fake watermark that appears to be a noise signal but has a high correlation with the distributed Work. Such a reference pattern can be built by extracting and distorting some features of the distributed Work. This reference pattern, which by definition is found in the distributed Work, is very likely found in the true original. The adversary subtracts the reference pattern from the distributed Work to create his or her fake original and then locks both his or her fake original and fake watermark in a safe.

---

[3] Sometimes called the *Craver attack* or the *IBM attack* after their introduction in [102] by Craver *et al.* of IBM.

In the following investigation, we illustrate an ambiguity attack on the E_BLIND embedder. We assume blind detection.

INVESTIGATION

Ambiguity Attack on a System with Blind Detection

As an example of this attack, consider the two images of Figure 10.7. The image in Figure 10.7(a) will be called the *true original*. This image has been watermarked with the E_BLIND embedding algorithm of System 1, and the resulting *distributed image* is shown in Figure 10.7(b). To create a fake watermark that has a high correlation with the distributed image but looks like noise, the adversary might replace the magnitude of the Fourier transform of the distributed image with random values. This signal, shown in Figure 10.8(a) after scaling to have unit length, has a correlation of nearly 17.0 with the distributed image. However, this signal does not look like a random noise pattern because the edges in the image are so dominant.

The edge structure in Figure 10.8(a) can be easily destroyed by introducing some random noise into the Fourier phase. We accomplished this by adding independent, random values to the image pixels before calculating the Fourier transform. Thus, the fake watermark shown in Figure 10.8(b) was generated via the following steps:

1. Add random noise to the distributed image (Figure 10.7b).

2. Calculate the Fourier transform of the resulting noisy image.

3. Scale the Fourier coefficients to have random magnitudes (unrelated to their original magnitudes).

4. Invert the Fourier transform and normalize to unit variance.

Even though this pattern looks like pure white noise, it has a correlation of 0.968 with the distributed image.

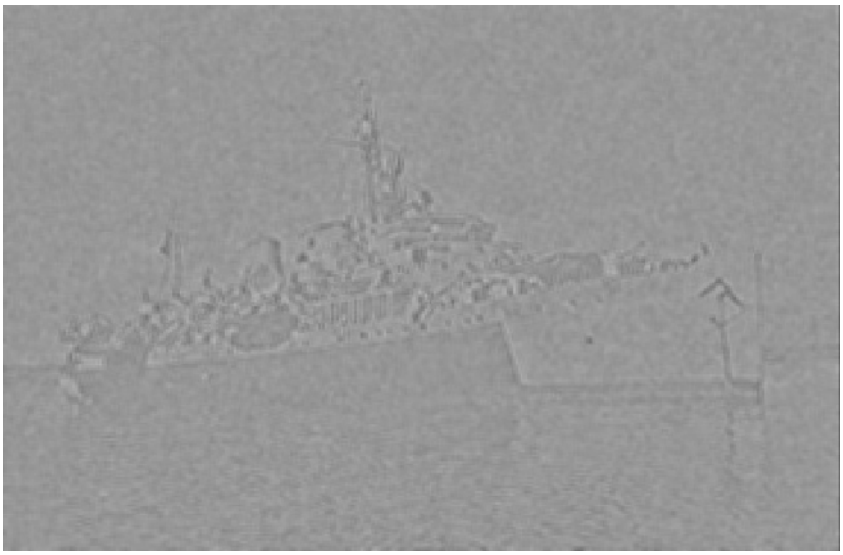
The adversary must now create his fake original. To do this he or she could simply subtract the fake watermark from the distributed image, but this would result in an original that is exactly orthogonal to the fake watermark. Although we would expect a random pattern to have a very low correlation with an original image, it is highly unlikely that the correlation would be exactly zero. Thus, the claims of the adversary would appear suspicious if his or her reference pattern were to have zero correlation with his or her original. To remedy this, the adversary constructs his or her fake original by subtracting only a portion of the fake watermark from the distributed image. The fake original, shown in Figure 10.9, was created by subtracting 99.5% of the fake watermark from the

(a)

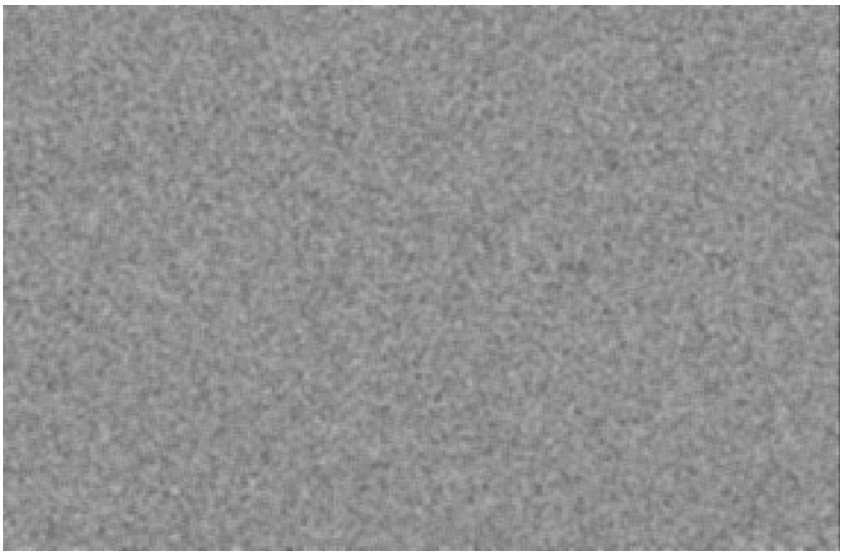
(b)

FIGURE 10.7

The original image (a) has been watermarked with the E_BLIND embedder using the white-noise reference pattern of System 1 to generate the distributed image (b).



(a)



(b)

FIGURE 10.8

Fake watermarks generated by (a) replacing the Fourier magnitudes of the distributed image with random noise and (b) further adding some random noise to the Fourier phase components.

FIGURE 10.9

The fake original constructed by subtracting 99.5% of the fake reference pattern from the distributed image.

distributed image. The results, presented in Table 10.3, show that watermark detection alone cannot be used to disambiguate the contradicting descendency claims of the true owner and the adversary.

Table 10.3 The correlation values, as reported by the D_LC detector, run on the original image, $\mathbf{c_o}$, the distributed image, $\mathbf{c_d}$, and the adversary's fake original, $\mathbf{c_f}$, for the owner's watermark, $\mathbf{w_r}$, and the adversary's fake watermark, $\mathbf{w_f}$. From these detection values, it is not possible to determine which of $\mathbf{c_o}$ and $\mathbf{c_f}$ is the true original.

	$\mathbf{c_o}$	$\mathbf{c_d}$	$\mathbf{c_f}$
$\mathbf{w_r}$	−0.016	0.973	0.971
$\mathbf{w_f}$	−0.968	0.970	0.005

Countering Ambiguity Attacks

Note that it is not possible to prevent an adversary from using this technique to create a fake original and a fake watermark. The primary defense against

this class of attacks is for the true owner of the Work to use a watermarking technique that can ensure that his or her original could not have been forged as previously described. The true owner can then argue that the evidence in support of his or her ownership claim is stronger than that of the adversarys.

The security hole being exploited by these ambiguity attacks has been called *invertibility* [102]. A watermarking scheme is invertible if the inverse of the embedding is computationally feasible. The embedding inverse is a function that acting on a distributed Work, $\mathbf{c_d}$, produces a fake original Work, $\mathbf{c_f}$, and a watermark, $\mathbf{w_f}$, such that the regular embedding function, $\mathcal{E}(\cdot)$, will embed the fake watermark into the fake original, resulting in the distributed Work. Symbolically, given the embedding function $\mathcal{E}(\mathbf{w},\mathbf{c})$, it is possible to create a function

$$\mathcal{E}^{-1}(\mathbf{w_f},\mathbf{c_d}) = \mathbf{c_f} \tag{10.10}$$

such that

$$\mathcal{E}(\mathbf{w_f},\mathbf{c_f}) = \mathbf{c_d}. \tag{10.11}$$

In the E_BLIND and E_BLK_BLIND algorithms of Chapter 3, the embedding function is the addition of a noise pattern. The inverse embedding function is simply the subtraction of that noise pattern.

Ambiguity attacks cannot be performed with noninvertible embedding techniques (i.e., those embedding techniques for which the inverse is not computationally feasible to devise). One approach for creating such a noninvertible embedder is to make the reference pattern dependent on the content of the original Work such that the reference pattern cannot be generated without access to the original. The use of one-way hash functions in this dependency ensures that the adversary cannot produce a fake original that will generate the correct fake watermark.

For example, consider generating a watermark reference pattern by seeding a pseudo-noise generator with a hash of the original image. An adversary may in reasonable time find a random watermark that has a high correlation with the distributed image. However, this random watermark would not be generated by a pseudo-random number (PN) generator seeded with a hash of the fake original. It is virtually impossible to find a watermark that is pseudo-randomly generated from a hash of the fake original and which has a high correlation with the distributed image.

10.3.5 **Sensitivity Analysis Attacks**

Sensitivity analysis is a technique used for unauthorized removal of a watermark when the adversary has a black-box detector. Described in [93, 217, 269], the sensitivity analysis attack uses the detector to estimate the direction that

yields a short path from the watermarked Work to the edge of the detection region. It is based on an assumption that the direction of a short path can be well approximated by the normal to the surface of the detection region and that this normal is relatively constant over a large part of the detection region. The correlation detection methods described in this book all satisfy these requirements.

The attack can be stated in three steps, illustrated in Figure 10.10 for a linear correlation detection region. The first step is to find a Work that lies very close to the detection region boundary, denoted Work A in the figure. This Work need not be perceptually similar to the watermarked Work (if it were, no further processing would be necessary). The boundary of the detection region can be reached in a number of ways by altering the watermarked Work. Some methods include decreasing the amplitude (contrast or volume) of the watermarked Work, replacing samples with the mean value of the Work, or building a linear combination of the watermarked Work and a different Work known to be unwatermarked. In all three cases, the distortion can be slowly increased until the watermark fails to be detected.

The second step is the approximation of the direction of the normal to the surface of the detection region at Work A. This can be accomplished with an iterative technique (described in material to follow), which is at the heart of the sensitivity analysis attack. Once the normal has been estimated, the third step is to scale and subtract the normal from the watermarked Work.

Two methods of estimating the N-dimensional normal vector have been proposed. In Linnartz and van Dijk [269] the vector is found by considering the effects of N independent modification vectors. Each of these is separately added to or subtracted from Work A with increasing amplitude until the detector ceases to detect the watermark. The normal to the surface is then approximated by the weighted sum of the N modification vectors, in which each is weighted by the negative of the corresponding scale factor found.

In Kalker *et al.* [217] the technique for estimating the normal to the detection region surface at Work A is iterative. For each iteration a random vector is added to Work A and the detection decision recorded. If the resulting Work causes a positive detection, the random vector is added to the estimate of the normal. If the resulting Work does not contain the watermark, the random vector is subtracted from the estimate.

Note that the success of this attack relies on the assumption that the normal to the boundary of the detection region at some point, probably distant from the watermarked Work, can be used by the adversary to find a short path out of the detection region. This is the case for the two detection regions defined by placing thresholds on linear correlation or normalized correlation. A sensitivity analysis attack would not be successful if the curvature of the

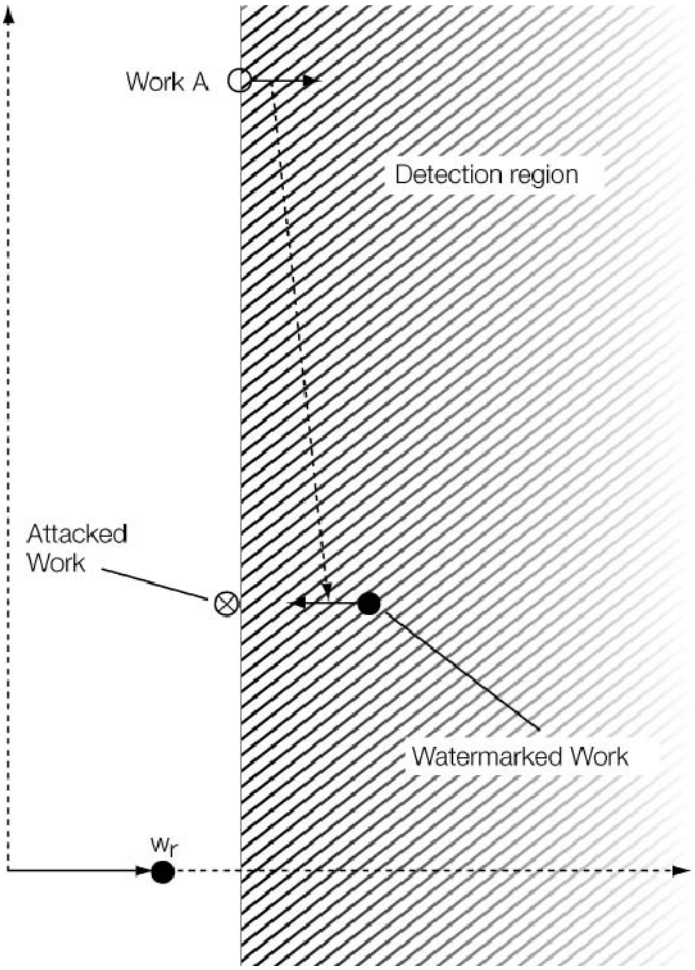


FIGURE 10.10

An illustration of the sensitivity analysis attack for a linear correlation detection region. Work A is a Work that lies on the boundary of the detection region. The normal to the surface of the detection region at Work A is estimated (pointing into the detection region), and the resulting vector is scaled and subtracted from the watermarked Work.

detection region boundary were such that the normal at one point on the surface provided little information about the direction of a short path. Construction of a detection region with this property remains an open research problem.

The following is an investigation in which we apply a sensitivity analysis attack to the D_LC detector.

INVESTIGATION

Sensitivity Analysis Attack

We begin with the same distributed image used in the previous Investigation, shown in Figure 10.7(b). The detector we will use is a modified version of the D_LC linear correlation detector, which outputs a 0 if the magnitude of the detection value is lower than a threshold of $\tau_{lc} = 0.7$ and outputs a 1 if the magnitude of the detection value is greater than or equal to the threshold.

To find an image on the detection region boundary, we have set random pixels of the image to neutral gray (equal to the mean of all pixel values) and have presented the resulting image to the detector. The detection decision changed from 1 to 0 after changing roughly 20,000 pixels. The boundary image, Work A (as shown in Figure 10.11), is the last image we obtained in which the detection decision was 1.

Next, we used the iterative method to estimate the normal to the surface of the detection region. In each iteration, a white-noise pattern with zero mean was

FIGURE 10.11

An example starting image, Work A, for the sensitivity analysis attack. This image is on the detection region boundary in that the watermark is detected in this image, but a slight change can cause the watermark to become undetected.

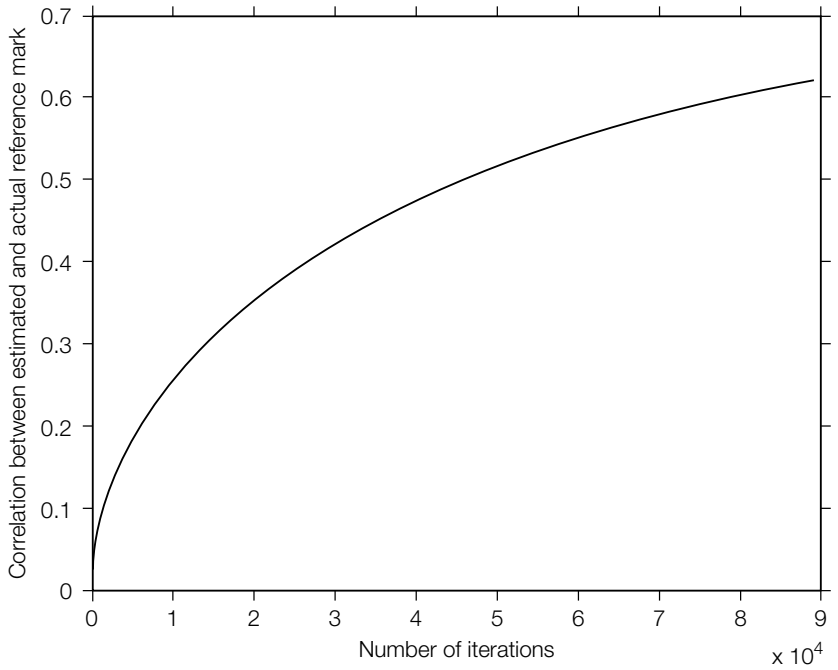


FIGURE 10.12

The correlation between the estimated normal vector and the actual watermark reference pattern as a function of the iteration number.

generated and added to Work A. The pattern was then subtracted from the estimated vector if it caused the Work to move out of the detection region; otherwise, it was added to the estimate. As the number of iterations increases, the estimate becomes more and more similar to the reference mark added to the image by the E_BLIND embedder. This can be seen in Figure 10.12, which plots the correlation between the estimated vector (after normalization to zero mean and unit variance) and the reference mark as a function of the number of iterations.

After nearly 90,000 iterations, the normalized correlation between the estimated normal vector and the reference vector is over 0.6. Although this is far from perfect, it is close enough that the watermark now can be removed with little impact on fidelity. The final step is to scale the estimated vector and subtract it from the watermarked Work. We used a simple binary search to find the minimum scaling factor required to make the watermark undetectable. The resulting image is shown in Figure 10.13.

FIGURE 10.13

The attacked image with watermark removed. This image has a detection value of 0.699, which is below the detection threshold.

10.3.6 Gradient Descent Attacks

In contrast to sensitivity analysis attacks, a *gradient descent* attack requires that the adversary have a detector that reports the actual detection values, rather than the final binary decision. The adversary uses the change in detection value, as the Work is slowly modified, to estimate the gradient of the detection statistic at the watermarked Work. It is assumed that the direction of steepest descent is the direction of a short path out of the detection region.

Figure 10.14 illustrates the attack for the case of detection by normalized correlation. In this figure, the reference pattern is a vector lying on the x-axis. The two white lines indicate the boundaries of the detection region. The shading indicates the normalized correlation at each point in the plane, with high values shown as white and low values as black.

Given a watermarked Work, we can use any search strategy to determine the local gradient of steepest descent. The Work can be moved along this path by some amount, and the process can iterate until the resulting Work falls just outside the detection region boundary.

The success of this attack relies on the assumption that the local gradient will point in the direction of a short path to the boundary. This is certainly the case for many detection statistics, including linear correlation and normalized correlation. To defend against this attack, the detection statistic within the detection region should not be monotonically decreasing toward the boundary.

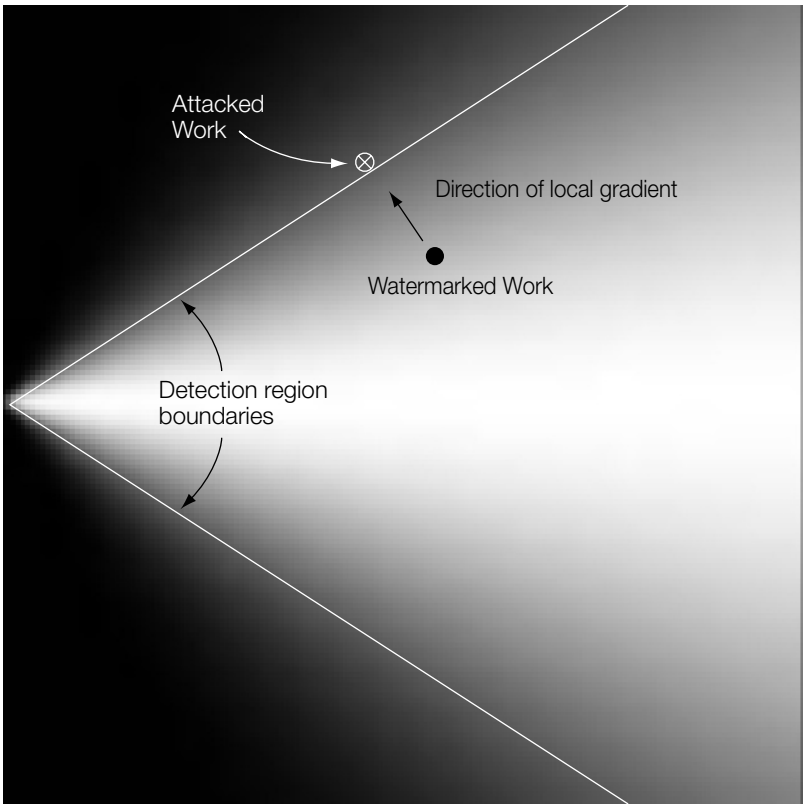


FIGURE 10.14

How a local gradient is used to find a short path out of the detection region.

Instead, it should contain many local minima such that the direction of the local gradient will not provide any information helpful in determining a short path out of the detection region.

10.4 SUMMARY

This chapter discussed the security of a watermark. Security is the resistance of a watermark to attempts by an adversary to defeat the purpose of the watermark. This is in contrast to the robustness of a watermark, which is its resistance to normal processing. The following main points were made in this chapter.

- The types of security required of a watermarking system vary with each application. Applications can require prevention of one or more of the following:
  - Unauthorized embedding.
  - Unauthorized detection.

  - Unauthorized removal.
  - System attacks.
- Within each of these types of restriction, varying degrees of security are required by an application.
- Many of the security requirements can be met by the incorporation of cryptographic tools, such as the following, to ensure the integrity of various aspects of the watermark message.
  - Encrypting embedded messages prevents unauthorized decoding.
  - Asymmetric-key cryptography or cryptographic signatures can be used to verify who embedded a watermark, thus preventing one form of unauthorized embedding.
  - Cryptographic signatures can be extended to verify that a watermark belongs in the cover Work in which it is found.
- The terms *private watermarking* and *public watermarking* do not imply any relationship to private-key and public-key cryptography.
- Private watermarking is not synonymous with informed detection.
- Many security requirements cannot be directly addressed with cryptographic tools. In general, such issues as the following continue to be open research problems:
  - Preventing unauthorized detection *without* decoding requires that the watermark must not change the statistical properties of the Work.
  - Preventing masking removal requires more sophisticated detectors that can detect and invert the masking distortion.
  - Preventing true removal requires preventing adversaries from identifying the boundary of the detection region. This is particularly difficult when the adversary has a detector.
- A number of known attacks were described. These were:
  - Scrambling attacks.
  - Pathological distortions.
  - Copy attacks.
  - Ambiguity attacks.
  - Sensitivity analysis attacks.
  - Gradient descent attacks.

CHAPTER 11

Content Authentication

In this chapter, we focus on how a watermark can be used to assist in maintaining and verifying the integrity of its associated cover Work. This is in contrast to Chapter 10, which is concerned with maintaining the integrity of the watermark itself.

The ease of tampering with photographs was demonstrated in Chapter 2 (Figure 2.3). Video and audio signals also can be easily altered. In many circumstances, alterations to content serve legitimate purposes. However, in other cases, the changes may be intentionally malicious or may inadvertently affect the interpretation of the Work. For example, an inadvertent change to an X-ray image might result in a misdiagnosis, whereas malicious tampering of photographic evidence in a criminal trial can result in either a wrong conviction or acquittal. Thus, in applications for which we must be certain a Work has not been altered, there is a need for verification or authentication of the integrity of the content. Specifically, we are interested in methods for answering the following questions:

- Has the Work been altered in any way whatsoever?
- Has the Work been *significantly* altered?
- What parts of the Work have been altered?
- Can an altered Work be restored?

Many nonwatermarking methods exist for answering these questions. For example, one method for answering the first question (i.e., has the Work been altered at all) is to append a cryptographic signature to it, as discussed in Appendix A. Even without cryptographic signatures, forensic specialists can often detect modifications to images and audio by identifying anomalies, such as missing shadows or discontinuities in background noise. Forensic analysis can also provide answers to determine where and how a Work was altered. The

problem of restoring corrupted content has received much attention from the image- and signal-processing communities, especially in the context of restoring content that has degraded with age. With all of these alternative approaches, why might we be interested in using watermarks?

There are two potential benefits to using watermarks in content authentication. First, watermarks remove any need to store separate, associated metadata, such as cryptographic signatures. This can be important in systems that must deal with legacy issues, such as old file formats that lack fields for the necessary metadata. A second potential advantage of watermarks is more subtle: A watermark undergoes the same transformations as the Work in which it is embedded. Unlike an appended signature, the watermark itself changes when the Work is corrupted. By comparing the watermark against a known reference, it might be possible to infer not just that an alteration occurred but what, when, and where changes happened.

These advantages must be weighed against the added complexity of using a watermark for authentication rather than appending a digital signature. Furthermore, there is a potentially serious adverse side effect of watermarking: Embedding a watermark changes a Work, albeit in a known and controlled manner. If we want to verify that Works are not changed, some applications may find even imperceptible alterations unacceptable.

Whether the advantages of watermarking outweigh the disadvantages probably depends on the application. However, it is not yet clear how important watermarks will become for verifying and maintaining the integrity of content. In many real applications of authentication, legacy issues may not be a problem. Moreover, the potential advantages of watermarks for localizing changes, identifying the type of alteration, and restoring the Work have yet to be fully realized.

The remainder of this chapter discusses issues involved in using watermarks to answer the four questions posed previously and describes a few basic methods that have been proposed. Section 11.1 deals with the question of simply determining whether or not a Work has been altered. Because a Work is considered altered if even 1 bit has been changed, we refer to this as *exact authentication*. In contrast, Section 11.2 discusses the question of determining whether a Work has been changed in a *significant* way. Here, we tolerate certain acceptable changes, such as flipping a single bit in a single pixel of an image, but try to detect unacceptable changes, such as removing a person from a photograph of a crime scene. We refer to this as *selective authentication*. Section 11.3 discusses the question of *localization* (i.e., identifying the time or location of those portions of a Work that have been altered, while verifying that other parts remain unchanged). Many authentication watermarks are designed to address this question. Finally, Section 11.4 discusses some research into the use of watermarks to aid in the *reconstruction* of Works that have been corrupted.

11.1 EXACT AUTHENTICATION

The most basic authentication task is to verify that a Work has not been altered *at all* since it left a trusted party. If even a single bit has been changed, the Work is regarded as inauthentic. By insisting on a perfect copy, we avoid any need to design algorithms that can differentiate between acceptable and unacceptable alterations. Each Work is just a collection of equally important bits.

We discuss two approaches to using watermarks for *exact* authentication. The first involves generating watermarks specifically designed to become undetectable when the Work is modified. The second involves embedding cryptographic signatures, which become invalid when the Work is modified.

The final topic of this section deals with a problem that arises in applications in which exact authentication is necessary; namely, that changes introduced to the cover Work by a watermark are sometimes unacceptable themselves. In such cases, a watermark might be embedded in a manner that allows exact removal or erasure, so that the original Work can be restored, bit-for-bit, during the authentication process.

11.1.1 Fragile Watermarks

A *fragile watermark* is simply a mark likely to become undetectable after a Work is modified in any way. Until now, we have considered fragility undesirable, seeking instead to design robust watermarks that can survive many forms of distortion. However, fragility can be an advantage for authentication purposes. If a very fragile mark is detected in a Work, we can infer that the Work has probably not been altered since the watermark was embedded. At least, it is unlikely the Work has been accidentally altered. Verifying that there has been no *malicious* alteration is discussed later.

A simple example of a fragile watermark is the least significant bit (LSB) watermark described in Chapter 5. Most global processes, such as filtering or lossy compression, effectively randomize the LSBs of a Work's representation. Thus, if a predefined bit sequence is embedded in the LSB plane of a Work, detection of this pattern implies that the Work has not undergone any such process.

In some applications, a different form of fragile watermark can be made by using an embedding method specific to the particular representation of the Work [454]. For example, several researchers have suggested methods of hiding information in halftoned representations of images [28, 156, 400, 434]. Another example of such a watermark, specific to the MPEG representation of video, is suggested in Linnartz and Talstra [267]. In MPEG, each frame of video is encoded in one of three formats, called I-, B-, and P-frames. Typically, MPEG encoders use regular sequences of frame types. However, there is no requirement that the coding sequence be regular, and it is possible to encode information by varying

the sequence of frame types. It is then possible to authenticate the received MPEG stream. However, the decoded baseband video cannot be reencoded as an authentic MPEG stream because the information regarding the order of I-, B-, and P-frames is lost in the decoding process. That is, the watermark does not survive MPEG decoding. If we assume that any change of the Work entails decoding and reencoding, the presence of such a watermark in the MPEG stream indicates that no change has occurred.[1]

Although fragile watermarks indicate that a Work has not been inadvertently modified, the use of predefined patterns cannot guarantee that no one has intentionally tampered with the Work. This is because an adversary can easily forge a fragile watermark if the embedded pattern is not dependent on the cover Work. In the case of an LSB watermark, forgery is a simple matter of copying the LSBs from the authentic Work to the tampered cover Work. In the case of an MPEG frame-type mark, the adversary can encode the tampered video with an encoder modified to use the same sequence of frame types as the original MPEG stream. Thus, the fragility of a watermark provides only limited authentication capabilities.

11.1.2 Embedded Signatures

The application of cryptographic authentication to images and other content is well understood. For example, Friedman [154, 155] proposed that "trustworthy cameras" be equipped with processors to compute an authentication signature that would be appended to the image. Of course, the authentication signature must be transmitted along with the associated Work. If the signature is transmitted as metadata within a header, there is a risk that this metadata may be lost, especially if the content undergoes a variety of format changes.

Watermarking can reduce the risk that the authentication signature is lost by embedding the signature within the cover Work. This permits format conversions to occur without the risk of losing the authentication information. Thus, representing the authentication signature as a watermark may allow the system to work with legacy systems that would otherwise have to be redesigned in order to guarantee that the header information is preserved.

Note that the signature may be embedded with either a robust or a fragile watermark. If the watermark is robust, the signature will (usually) be correctly extracted even if the Work has been modified. With a correct signature and a modified Work, it is extremely unlikely that the signature computed from the corrupted Work will match the correct signature. However, it is also extremely unlikely that a random signature will match a modified Work, and therefore

[1] It is debatable whether techniques such as the MPEG method mentioned here are watermarks in the strictest sense, in that they embed information in the representations of Works, rather than in the Works themselves.

the system will Work equally well with a fragile watermark. Because fragile watermarks are generally simpler than robust marks, they are typically preferred for embedding authentication signatures [449].

Of course, the very process of embedding a watermark alters the Work, causing the subsequent authentication test to fail. To prevent this, it is usually necessary to partition the Work into two parts, one of which is to be authenticated and the other to be altered to accommodate a watermark.

A simple example is to partition a Work such that the LSB plane holds the authentication signature computed from the remaining bits of the Work. Note that in this arrangement we only need a small number of bits, say 160, to hold the signature. Thus, all but 160 bits of the Work can be authenticated. If desired, the specific 160 bits used can be determined from a key shared between the sender and receiver.

11.1.3 Erasable Watermarks

In some applications, even the minimal distortion introduced by embedding a 160-bit signature might be unacceptable. For example, in medical applications, *any* modification of an image might become an issue in a malpractice suit. It might be easy to convince an engineer that such an embedded signature will not change a doctor's interpretation of an image, but will the same arguments be persuasive in a courtroom? In such circumstances, the only way to guarantee that no significant change has occurred is for there to be no change at all. This has led to an interest in using *erasable watermarks*[2] for authentication (i.e., watermarks that can be removed from their associated cover Works to obtain exact copies of the original unwatermarked Works).

If an erasable watermark can be constructed, there is no problem with cryptographic signatures becoming invalid after embedding. The following steps outline a general protocol:

1. The originator of a Work computes a signature using *all* information in the Work. The signature is then embedded in an erasable manner within the Work.
2. The recipient of the Work extracts and records the embedded signature.
3. The recipient erases the watermark from the cover Work. At this point, the Work should be identical to the original unwatermarked Work.
4. To verify this, the recipient computes a one-way hash of the Work and compares this to the hash decoded from the sender's signature.

[2] These watermarks have also been called *invertible* [140]. However, we prefer not to use this term because it has a conflicting meaning in the context of ambiguity attacks [102].

5. If and only if the computed and retrieved hashes are identical is the received cover Work authentic.

The design of an erasable watermark poses a fundamental problem. In brief, it is theoretically impossible to make an erasable mark that can be embedded in 100% of digital content. We discuss this problem in the following, and illustrate its impact with a simple investigation. We then describe a general method of working around this problem. This method results in erasable watermarks that can be embedded in all content *likely* to arise in a given application, and can thus be used in practice.

Fundamental Problem with Erasability

Ideally, an erasable watermarking system for authentication would be able to (1) embed a signature with 100% effectiveness, (2) restore the watermarked Work to its original state, and (3) have a very low false positive probability. Unfortunately, when marking digital media, achieving the latter two of these ideal requirements makes it impossible to achieve the first.

Digital Works are typically represented with a fixed number of bits. Thus, there is a fixed, finite, albeit very large, set of possible Works. For example, each of the images used for the experiments in this book is represented by 240 × 368 pixels, and each pixel is represented by 8 bits. Thus, there are a total of $2^{706,560}$ possible images in the space of original unwatermarked images.

Erasability requires that the original Work can be recovered from the watermarked Work. This means that no two unwatermarked Works entered into the embedder can result in the same watermarked Work. That is, each unwatermarked Work must map to its own, unique watermarked Work. Thus, if the embedder can successfully embed a watermark into every possible Work (i.e., every one of the $2^{706,560}$ possible images), the number of *watermarked* Works must be equal to the number of *possible* Works (i.e., there must be $2^{706,560}$ distinct watermarked images). If watermarked Works are represented with the same number of bits as unwatermarked Works, this means that *all* Works must be considered watermarked. Therefore, the only way to achieve 100% effectiveness is to allow for 100% false positives.

Note that this problem does not arise with nonerasable watermarking systems. In such systems, we can allow many unwatermarked Works to map into each watermarked Work, and thus the number of watermarked Works can be a small fraction of the total number of possible Works. For example, consider LSB watermarking, in which we replace the LSBs of a Work with a given message. Using this system, any pair of unwatermarked Works that differ only in their LSBs will result in the *same* watermarked Work. Thus, if there are N samples or pixels in each Work, 2^N different Works map into each watermarked Work, and only 1 in 2^N possible Works needs to be considered watermarked. The following investigation illustrates the problem with a simple form of erasable watermark.

INVESTIGATION

Watermarking with Modulo Addition

To illustrate how an erasable watermark might be constructed, and how the fundamental problem with embedding effectiveness is manifested, we describe an algorithm for embedding an erasable watermark in 8-bit images. We then examine the conditions under which the watermark embedder will fail.

System 16: E_MOD/D_LC

We denote our erasable watermark by E_MOD/D_LC. It is a simple modification of the E_BLIND/D_LC system. At first glance, the E_BLIND embedding algorithm might seem already to serve our purpose. That algorithm computes the marked image, $\mathbf{c_w}$, as

$$\mathbf{c_w} = \mathbf{c_o} + \alpha \mathbf{w}_m \tag{11.1}$$

$$\mathbf{w}_m = \mathbf{w_r}, \tag{11.2}$$

where $\mathbf{c_o}$ is the original image, $\mathbf{w}_m$ is a watermark message pattern, $\mathbf{w_r}$ is a watermark reference pattern, and α is a strength parameter. Suppose that a predefined value of α is always used during embedding, so that the recipient of the image knows this value *a priori*. After determining the message pattern, $\mathbf{w}_m$, that was embedded, the recipient should be able to remove the mark by computing

$$\mathbf{c}'_\mathbf{o} = \mathbf{c_w} - \alpha \mathbf{w}_m. \tag{11.3}$$

In principle, this should yield an exact copy of $\mathbf{c_o}$.

However, Equation 11.2 cannot be exactly implemented in practice, in that adding the reference pattern might result in pixel values that lie outside the 8-bit range of 0 to 255. That is, the output of this equation might not be a member of the set of $2^{706,560}$ possible images. In our initial implementation of the E_BLIND algorithm, we solved the problem by clipping all pixel values to the range of 0 to 255. However, because the recipient of the image does not know which pixels are clipped, there is no way to recover the exact original.

One solution to the problem of clipping is to restrict the intensity range of input images to between, say, 10 and 245. Then, assuming that the amplitude of the added watermark pattern does not exceed 10, no underflow or overflow will occur and clipping will be avoided. Unfortunately, in many applications we will not be able to enforce this constraint.

Honsinger *et al.* [200] propose solving this problem by the use of modulo arithmetic, rather than through clipping. Thus, the addition of, say, 5 to an 8-bit value of 253 does not result in clipping at 255 but in a value of 2. Similarly, subtracting 5 from the value 2 results in the original value of 253. As long as a large enough fraction of the pixels does not wrap around, the D_LC detector should be able to detect the watermark. The E_MOD embedding algorithm

computes each pixel of the watermarked image as

$$\mathbf{c_w}[x, y] = (\mathbf{c_o}[x, y] + a\mathbf{w}_m[x, y]) \bmod 256 \tag{11.4}$$

Of course, this wraparound process can produce perceptible artifacts in the form of "salt-and-pepper" noise. This is illustrated in Figure 11.1, in which we have used the E_MOD algorithm to embed a white-noise watermark pattern. Where the pattern caused pixel values to wrap around, the pixels appear as isolated black or white dots.

Although salt-and-pepper artifacts might appear ugly, it must be remembered that they will be removed when the original Work is recovered. Thus, whether the salt-and-pepper noise is important depends on what will be done with the watermarked Works prior to erasing the watermark. For example, suppose the watermarked Works are used only for browsing. Once a user has identified a Work of interest, he or she authenticates it and reconstructs the original before proceeding to use it for other tasks. In such a scenario, the watermarked Works need only be recognizable, and salt-and-pepper artifacts might not be a serious problem.

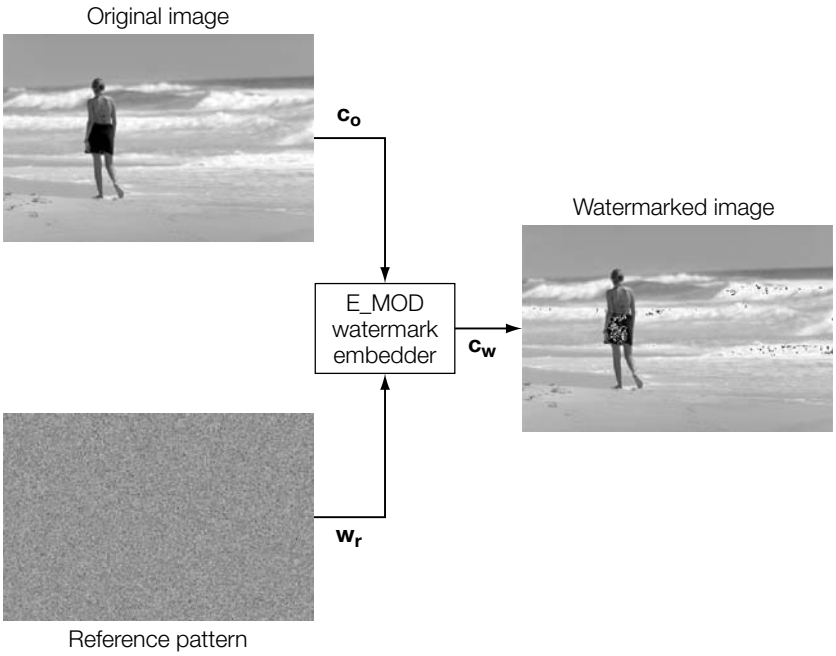


FIGURE 11.1

Effect of embedding a watermark with the E_MOD embedding algorithm. Note the salt-and-pepper noise apparent in the darkest and lightest areas.

Experiment 1

The salt-and-pepper noise also adversely affects watermark detection. However, for most of the images in our test set, its impact is not too severe. This is shown by the results of our first experiment, in which the E_MOD algorithm was used to embed two different watermarks in each of 2,000 images, one representing a message of $m = 0$ and the other representing $m = 1$. The resulting frequencies of different detection values, as measured by the D_LC detector, are shown in Figure 11.2. The embedder succeeded in embedding the mark most of the time.

Experiment 2

We know that there must be large classes of images for which the E_MOD embedder will fail. Can we characterize these? Clearly, the embedder is likely to fail on images whose pixel values are predominantly close to the extremes of the allowable range. Thus, we would not wish to use this algorithm to authenticate black-and-white line drawings, in which each pixel is either 0 or 255. However, it is also likely to fail on images whose histograms are relatively flat.

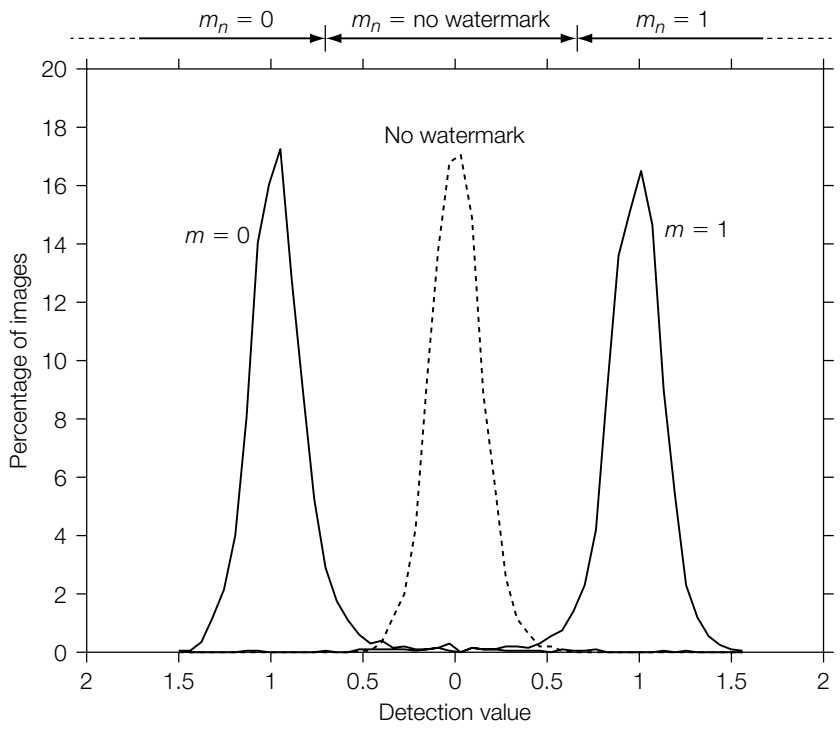


FIGURE 11.2

Test results for E_MOD/D_LC watermarking system.

The problem with a flat histogram is that the distribution of pixel values is unchanged when something is added to it modulo 256. Correlation-based detection depends on the fact that for any element of the reference pattern, $\mathbf{w_r}[x, y]$, the expected value of $\mathbf{w_r}[x, y](\mathbf{c_o}[x, y] + \mathbf{w_r}[x, y])$ is greater than the expected value of $\mathbf{w_r}[x, y]\mathbf{c_o}[x, y]$, with $\mathbf{c_o}[x, y]$ drawn randomly from the distribution of pixel values in the image. This is true regardless of the distribution of $\mathbf{c_o}[x, y]$.

Whether the same relation holds when addition is modulo 256—that is,

$$E(\mathbf{w_r}[x, y](\mathbf{c_o}[x, y] + \mathbf{w_r}[x, y]) \bmod 256)) > E(\mathbf{w_r}[x, y]\mathbf{c_o}[x, y]), \tag{11.5}$$

is dependent on the distribution of $\mathbf{c_o}[x, y]$. In particular, when $\mathbf{c_o}[x, y]$ is uniformly distributed between 0 and 255, $(\mathbf{c_o}[x, y] + \mathbf{w_r}[x, y])$ mod 256 is also uniformly distributed between 0 and 255. Thus, the two expected values are the

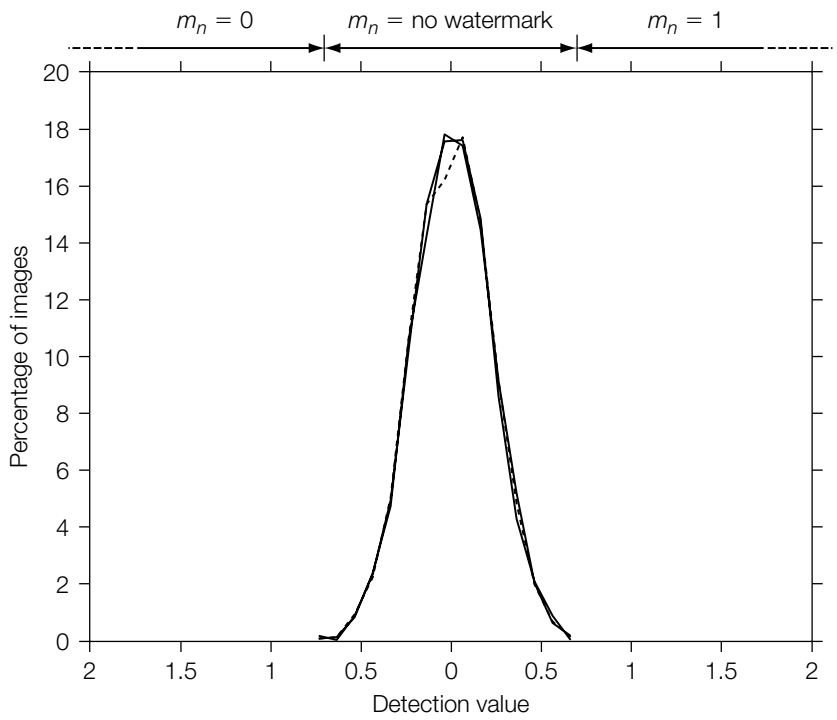


FIGURE 11.3

Test results for E_MOD/D_LC watermarking system when used on images with equalized histograms.

same and, on average, the embedding process should have no effect on the detection value.

This was verified with our second experiment, which was a repeat of the first except that prior to watermark embedding we applied histogram equalization, as described in Gonzalez and Woods [167]. Histogram equalization is a common image enhancement technique that adjusts pixel values so that the resulting image histogram is flat. The results are shown in Figure 11.3. Here there is essentially no difference between the distribution of detection values obtained from marked and unmarked images, and therefore all three curves lie on top of one another. The embedding system failed completely.

Algorithms such as E_MOD must be used with great care. In applications for which we can be sure that the vast majority of pixel values will be far from the two extremes of 0 and 255, watermarks can be embedded reliably. If, however, images are likely to be saturated, such as black-and-white line drawings, or processed by histogram equalization, the algorithm is likely to fail, and a different approach will be required.

Practical Solutions for Erasability

The previous section and accompanying Investigation illustrated how sensitive the property of erasability can be to the distribution of Works. The biggest challenge for designers of erasable watermarks is to achieve high erasable embedding capacity with low embedding distortion for as wide a class of Works as possible.[3] The most successful approaches are variations of several different basic design principles that can be broadly divided into three classes: methods based on lossless compression of image components [68, 140, 165], *difference expansion* [16, 183, 384, 408], and *histogram shifting* [310, 418, 420, 459].

The first class of methods is conceptually rather simple. The encoder starts by identifying a subset A of the Work that is losslessly compressible and that can be "randomized" without introducing a large perceptual distortion. The set A is first compressed to a datastream $C(A)$ and this compressed bitstream is then concatenated with the watermark payload m. This composite payload is then embedded in the Work by replacing A with the concatenation of m and $C(A)$. The decoder extracts both the payload m and the compressed bitstream $C(A)$. The original copy of the Work is obtained by decompressing $C(A)$ and inserting it back into the watermarked Work. The class of Works for which this erasable watermarking algorithm works is easily describable as the set of all Works that have such a compressible component A.

[3] Information-theoretical bounds expressing the tradeoff between distortion and erasable capacity have been obtained by Kalker and Willems [218].

As a simple example, consider grayscale images with A being the fourth LSB plane. Figures 11.4 and 11.5 show the original image, Lena, and its fourth LSB plane. As can be seen, the fourth bit plane contains a visible structure that is compressible using standard lossless image compression schemes, such as CALIC or JBIG2 [359]. Since the fourth LSB plane of most natural images is losslessly compressible, with the exception of extremely noisy images, and randomization of this bit plane usually does not introduce a large perceptual distortion, most natural images can be watermarked using this erasable technique. More advanced versions of this method do not work directly with a subset of images but instead extract a losslessly compressible "feature vector" through a transformation that makes better use of spatial correlations among

FIGURE 11.4

Original 512 × 512 grayscale image of Lena.

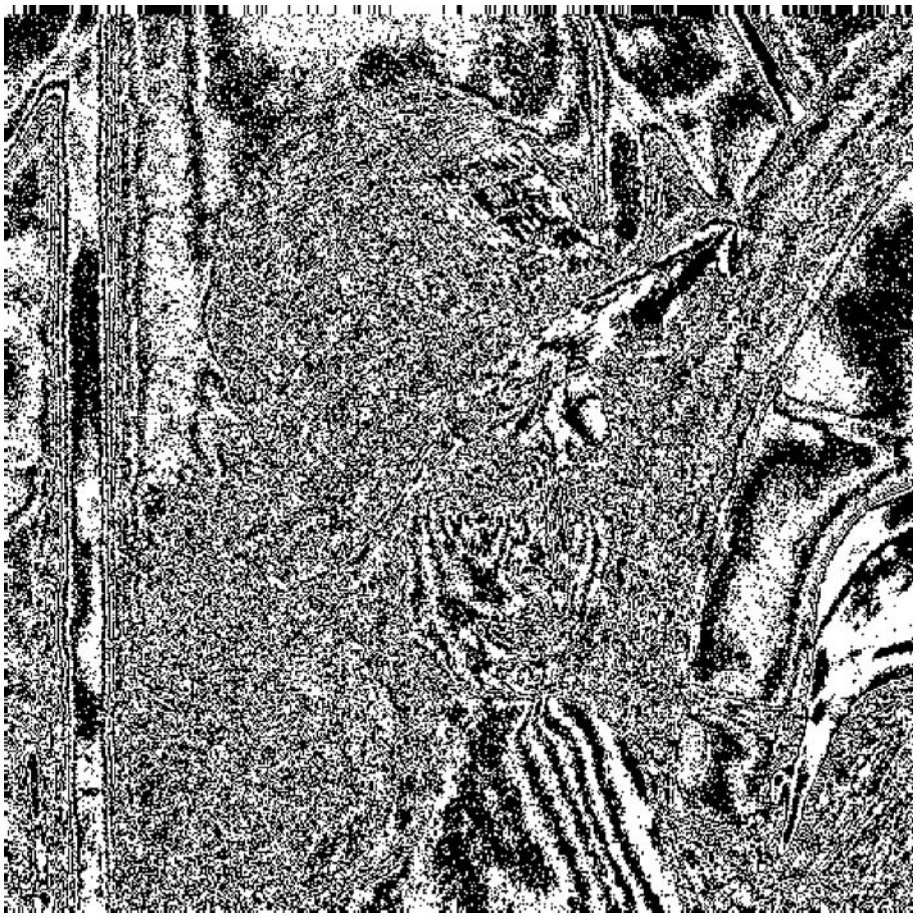


FIGURE 11.5

The fourth LSB plane of Lena.

pixels [68, 165]. The benefit is increased payload with decreased embedding distortion.

Erasable watermarks based on difference expansion use the fact that neighboring pixels in images are more likely to have similar values rather than very different values. Consequently, the difference between two neighboring pixels has a smaller dynamic range than the pixel values themselves and thus can be "expanded." This expanded difference then can be modified to encode the watermark [15, 16, 408].

Consider transforming a pair of neighboring pixels (x_1, x_2) with $x_1, x_2 \in \{0, \ldots, 255\}$ to the pair

$$(y_1, y_2) = T(x_1, x_2) = (2x_1 - x_2, 2x_2 - x_1).$$

If x_1 and x_2 are "similar," the transformed pair (y_1, y_2) will fall with high probability into the same dynamic range. For example, for the pair (59, 54), we would obtain the transformed pair $T(59, 54) = (64, 49)$. Note that the difference $y_1 - y_2$ is always a multiple of 3 because $y_1 - y_2 = 2x_1 - x_2 - (2x_2 - x_1) = 3(x_1 - x_2)$. Thus, we can embed bit 1 into the transformed pair by adding 1 to y_1. By subtracting 1 from y_1, we embed a 0. The decoder will read the bit by calculating $d = y_1 - y_2 \bmod 3$ with the convention that $d = 1$ encodes a 1 and $d = 2$ encodes a 0. The original pixel pair is obtained by rounding y_1 to the closest multiple of 3 and inverting the transformation T:

$$(x_1, x_2) = T^{-1}(y_1, y_2) = \left(\frac{4y_1 + 2y_2}{6}, \frac{4y_2 + 2y_1}{6} \right).$$

Continuing with our example, let us suppose that we are embedding a 0. The transformed pair $T(59, 54) = (64, 49)$ is thus changed to (63, 49). If we were to embed a 1, we would change 64 to 65 instead of 63. The decoder first calculates $d = (63 - 49) \bmod 3 = 2$, which corresponds to watermark bit 0. The original pixel pair is extracted by first finding the closest value to 63 that gives a difference divisible by 3, which is 64 because $64 - 49 = 15$, and then evaluating $T^{-1}(64, 49) = (\frac{464 + 249}{6}, \frac{449 + 264}{6}) = (59, 54)$. To finish the description of the method, however, we need to specify what should be done if the transformed pair or its modified version fall out of the pixel dynamic range (outside of the set $\{0, \ldots, 255\}$). We describe this for a more general version of this erasable watermarking scheme.

INVESTIGATION

Erasable Watermarking Based on Difference Expansion

Starting with an integer $n > 0$, consider the mapping T defined as

$$(y_1, y_2) = T(x_1, x_2) = ((n + 1)x_1 - nx_2, (n + 1)x_2 - nx_1).$$

The scheme described previously is obtained for $n = 1$. Note that the difference $y_1 - y_2 = (2n + 1)(x_1 - x_2)$ is always a multiple of $2n + 1$. A pixel pair (x_1, x_2) is called *embeddable* if both values in the pairs $(y_1 - n, y_2)$ and $(y_1 + n, y_2)$ are within the dynamic range $\{0, \ldots, 255\}$. At each embeddable pixel pair, we embed a symbol $s \in \{1, 2, \ldots, 2n\}$ by modifying y_1 by at most n so that the difference $d = (y_1 - y_2) \bmod (2n + 1) = s$. We reserve the case $d = 0$ to mark nonembeddable pixel pairs. If a pixel pair is not embeddable, it is not transformed. Instead, x_1 is modified by adding a correction term c to it so that $x_1 + c - x_2$ is a multiple of $2n + 1$ *and* the modified pair is within the dynamic range. Provided x_1 is at least n

away from the boundary of the dynamic range (i.e., $n \le x_1 \le 255 - n$), this can be always achieved with exactly one $c \in \{-n, \ldots, n\}$. We explain later how to treat the cases when $x_1 + c$ falls out of the dynamic range. The sequence of corrections c must be added to the watermark payload and also embedded in the image in expandable pairs so that the decoder can recover the original values of nonembeddable pairs. Note that the decoder can identify all nonembeddable pairs by checking whether $(y_1 - y_2)$ is a multiple of $(2n + 1)$.

The decoder first extracts the payload and the corrections from all pixel pairs (y_1, y_2) for which $(y_1 - y_2) \bmod (2n + 1) \neq 0$, skipping the remaining pairs as they do not contain any embedded data. Then, the decoder goes through the Work once more and subtracts the extracted correction term from the first element of each nonembeddable pair to obtain the original Work.

The payload the encoder needs to embed is composed of the message *and* the sequence of corrections c needed to reconstruct the original Work. Because most pixel pairs in a typical image will be embeddable, the sequence of corrections typically will be short, which will provide a relatively large space for the watermark payload. If p is the number of all pixel pairs in the Work and e the number of all embeddable pairs, the maximium size of payload that can be embedded is $e \log_2 2n - (p - e) \log_2(2n + 1)$. This is because we need $\log_2 2n$ bits to encode one symbol of the watermark payload and $\log_2(2n + 1)$ bits to encode each correction. Typically, $p - e \ll p$ and the capacity b expressed in bits per pixel is approximately $b = \frac{\log_2 2n}{2}$bpp.

As already hinted above, the encoder might encounter a situation when $x_1 + c \in 0, \ldots, 255$, which can only happen when either $x_1 < n$ or $x_1 > 255 - n$. Whenever $x_1 + c$ falls out of the dynamic range, we modify x_1 as much as we can to 0 or 255 by adding $c_1 = -x_1$ or $c_1 = 255 - x_1$ to x_1 *and* we add $c_1 - c$ to x_2. Note that the difference $d = x_1 + c_1 - (c_1 - c + x_2) = x_1 + c - x_2$ becomes a multiple of $2n + 1$ as required.[4] The encoder must embed both corrections c_1 and $c_1 - c$ whenever either $x_1 = 0$ or $x_1 = 255$ so that the decoder can recognize the case when two correction terms were applied to the pair. The rest of the algorithm remains the same.

One important advantage of erasable watermarking methods based on difference expansion is that the embedding distortion approximates sharpening (i.e., high-pass filtering), and the watermarked image looks acceptable even with a low PSNR. Figures 11.6 and 11.7 show the original 512 × 512 test image, Lena, and its watermarked version obtained by embedding the maximum payload using $n = 1$. The erasable watermark capacity is $b = 0.49$ bpp or 512 × 512 × 0.49 = 128,450 bits at a PSNR of 30.01 dB. Figures 11.8 and 11.9 show the fully embedded image, the watermark payload, and embedding distortion for $n = 2$ and $n = 3$.

[4] By listing all possibilities, the reader can easily verify that provided $n \le 5$, then $(c_1 - c + x_2)$ must always be within the required dynamic range.

FIGURE 11.6

Original 512 × 512 grayscale image of Lena.

FIGURE 11.7

Fully embedded, $n = 1, b = 0.49$ bpp, and PSNR = 30.01 dB.

FIGURE 11.8

Fully embedded, $n = 2, b = 0.98$ bpp, and PSNR = 25.33 dB.

FIGURE 11.9

Fully embedded, $n = 3, b = 1.21$ bpp, and PSNR = 23.15 dB.

Another large class of simple erasable watermarking schemes that provide good capacity with low embedding distortion is based on modifying the histogram of an image before embedding [310, 418, 420, 459]. Intuitively, if the pixel values in a grayscale image do not cover the whole dynamic range, then an erasable watermark can be embedded by using the missing grayscale values. If no unused values exist, one can manufacture gaps in the histogram before applying this idea. The method described below appeared in Neuschmied *et al.* [310].

Denoting by $\mathbf{h}[i], i = 0, \ldots, 255$, the histogram of a grayscale image, let us assume first that the histogram contains an empty bin with index i_e, so that $\mathbf{h}[i_e] = 0$. We first locate the most frequent value in the histogram, i_{max}. Without loss of generality, let us assume that the most frequently occuring gray-level value, i_{max}, is darker than the gray-level value, i_e (i.e., $i_{max} < i_e$). If we now add 1 to all gray-level values in the range i_{max} to i_e, then the histogram, $\mathbf{h}'$, is such that, $\mathbf{h}'[i] = \mathbf{h}[i]$ for $i = 1, \ldots, i_{max} - 1$, and for $i = i_e + 1, \ldots, 255$, since these values are untouched. We also have that $\mathbf{h}'[i_{max}] = 0$, since all pixels that formerly had this value have now been incremented by one. For those gray-level values in the range $i_{max} < i \leq i_e$, we have $\mathbf{h}'[i] = \mathbf{h}[i - 1]$. This is achieved by adding 1 to all pixels in the Work that have gray-level values in the range $[i_{max}, i_e - 1]$. The histogram value $\mathbf{h}'[i_{max}]$ equals zero. In this new image, we can embed an erasable watermark consisting of $\mathbf{h}'[i_{max}]$ bits simply by reserving the value i_{max} for 0 and $i_{max} + 1$ for 1. Note that in the watermarked image, we know that gray-level values i_{max} and $i_{max} + 1$ were originally equal to i_{max} in the unwatermarked image.

The decoder extracts the message by following the same embedding path through the watermarked Work, extracting the bits from grayscale values i_{max}, $i_{max} + 1$. After extraction, all pixels with values in the set $\{i_{max}, i_{max} + 1\}$ are changed to i_{max}, and all pixels with values $i_{max} + 1 < i \leq i_e$ are decreased by 1. This restores the original Work. The capacity of this scheme is $\mathbf{h}[i_{max}]$ bits and can be increased by applying the same process again to the embedded image.

This method can be extended to work for all images, including those with a full dynamic range when all histogram bins are nonempty (see Figure 11.10). Instead of locating an empty histogram bin, the encoder finds the minimum and maximum of the histogram, say at indices i_{min} and i_{max}. Suppose for simplicity that $i_{max} < i_{min}$. The encoder can manufacture an empty bin at i_{min} by replacing all grayscale values i_{min} with $i_{min} - 1$. The histogram of the modified image will thus have a gap at i_{min}, $\mathbf{h}[i_{min}] = 0$ (Figure 11.11). To be able to reconstruct the original Work, the information about positions of pixels with values i_{min} must be added in a compressed form to the watermark payload. This location information could be captured, for example, by scanning the image row by row and assigning a bit 1 to pixels with values i_{min} and 0 to the remaining pixels. The length of the resulting binary vector $\mathbf{v}$ is equal to the number of pixels. Since $\mathbf{v}[i] = 0$ for the vast majority of pixels, $\mathbf{v}$ will be losslessly compressible to $|C(\mathbf{v})|$ bits. Since the histogram now contains an empty bin, we can apply the method described in the previous paragraph with the only difference that

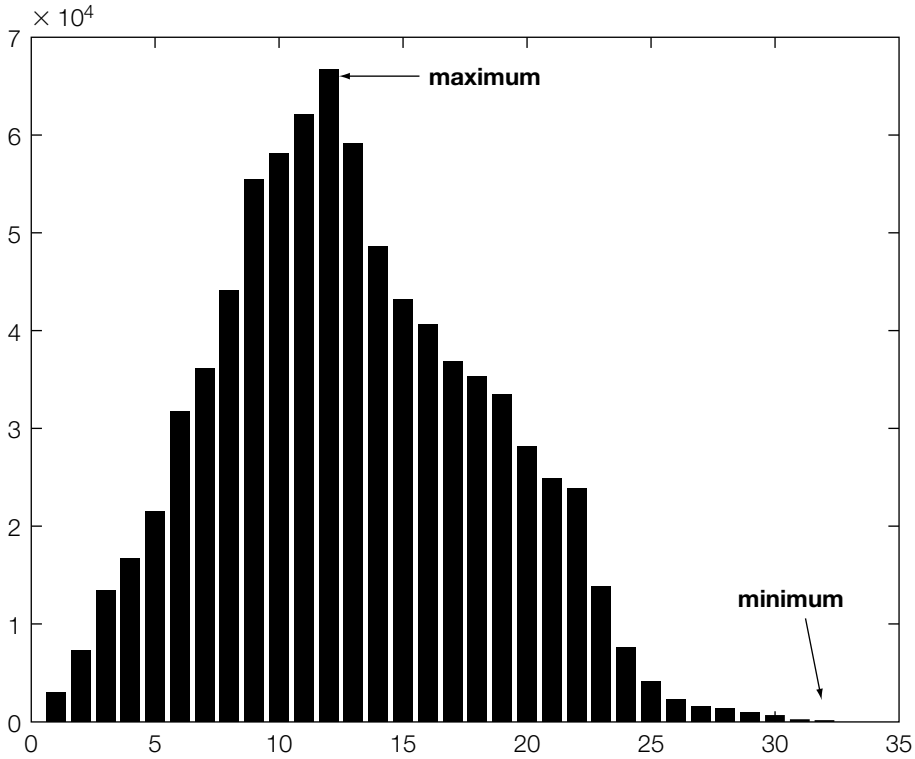


FIGURE 11.10

Original histogram.

now we have to embed $|C(\mathbf{v})|$ more bits together with the watermark payload. Thus, the capacity of this erasable watermarking method is $\mathbf{h}[i_{max}] - |C(\mathbf{v})|$.

The decoder extracts the message and the location information in the same way as before and reconstructs the image with an empty histogram bin, $\mathbf{h}[i_{min}] = 0$. Pixels with location indices obtained from the extracted combined payload are then set to i_{min} to recover the original Work. This method can be applied repeatedly to increase the capacity.

This method requires communication of some overhead—the location of the maximum and the minimum (gap) in the histogram, which would be 16 bits for a grayscale image. This can be done in several different ways. For example, 16 pixels from the Work can be put aside using a secret key. These pixels will hold the overhead bits and will not participate in the erasable scheme. The overhead is then embedded in a noninvertible manner in the LSBs of the reserved pixels. To reconstruct the original, the original 16 LSBs need to be added to the total payload.

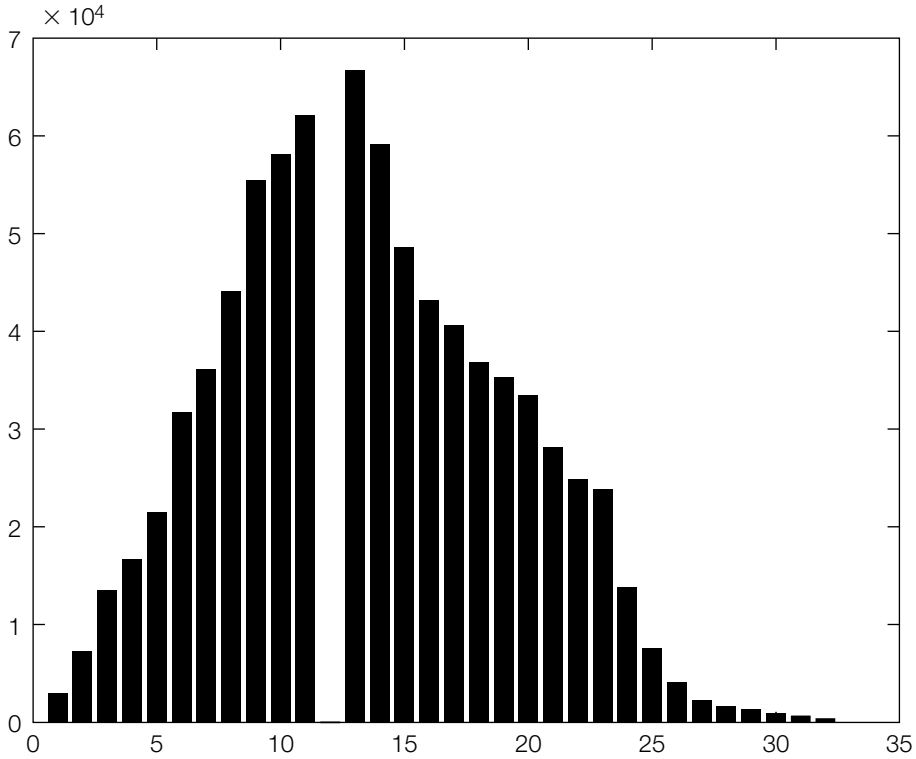


FIGURE 11.11

Histogram with a gap created by shifting the segment between the maximum and minimum histogram values.

This method works best with Works whose histogram has a narrow dynamic range and exhibits a sharp maximum, such as Works represented in some transform domain [418, 459].

Erasable watermarking schemes with high capacity enable construction of "robust" erasable schemes. In Investigation "Watermarking with Modulo Addition," we explained that clipping is the reason why additive robust watermarking schemes are in general not erasable (i.e., it is not possible to obtain the original Work, $\mathbf{x}$, from its watermarked version, $\mathbf{y} = \mathbf{x} + \mathbf{w}$, simply by subtracting $\mathbf{w}$ from $\mathbf{y}$). This problem can, however, be solved by embedding the clipped values together with their locations as an erasable watermark into the watermarked Work, $\mathbf{y}$, thus obtaining the twice-watermarked Work, $\mathbf{y}'$. The original Work can be recovered by first extracting the erasable payload from $\mathbf{y}'$, then "unclipping" the values of the watermarked Work, $\mathbf{y}$, before finally subtracting the watermark pattern, $\mathbf{w}$, to obtain the original Work, $\mathbf{x}$. To detect the watermark, $\mathbf{w}$, in the twice-watermarked Work, $\mathbf{y}'$, the additive watermarking scheme must be robust to the distortion introduced by the erasable watermark.

The advantage of combining the additive watermark with an erasable scheme is that if the watermarked image is not attacked, one can recover the exact copy of the original and thus undo any distortion due to insertion of the watermarks. Conversely, if the watermarked Work is attacked, the robust watermark may be still detectable, even though the erasable watermark cannot be recovered. Specific examples of robust erasable watermarking schemes include Vasudev and Ratnakar [428] and Coltuc and Chassery [83].

11.2 SELECTIVE AUTHENTICATION

Exact authentication is appropriate in many applications. For example, a change of just one or two characters in a text message—a handful of bits—can result in a substantially different meaning. However, in an image or an audio clip, a change of a couple bits rarely makes a difference of any importance. In fact, such distortions as lossy compression can change many bits in a Work, with the express intention of making no change in its perceptual quality. Thus, even though the two images of Figure 11.12 appear identical, (a) is a JPEG-compressed version of (b). In many applications, the perceptual similarity between images suggests that the compressed version is authentic. However, the image in Figure 11.12(b) will fail to pass an exact authentication test. This leads to a desire to develop tools that can perform *selective* authentication, in which only *significant* changes cause authentication to fail.

We begin this section with a discussion of possible requirements for a selective authentication system. The central issue here is deciding what types of transformations or distortions are significant enough to cause authentication to fail. We then discuss three basic approaches to building such systems. The first two approaches, *semi-fragile watermarks* and *semi-fragile signatures,* parallel the two approaches to exact authentication discussed in the preceding section. The third approach, *telltale watermarks,* is potentially more interesting, as it points toward systems that can identify the specific transformations a Work has undergone, rather than simply saying whether or not the Work has been significantly altered.

11.2.1 Legitimate versus Illegitimate Distortions

The requirements of a selective authentication system can be expressed in terms of distortions that might be applied to a Work, such as lossy compression, filtering, editing, and so on. We divide these distortions into two groups: *legitimate distortions* and *illegitimate distortions*. When a Work undergoes a legitimate distortion, the authentication system should indicate that the Work is authentic. Conversely, when a Work undergoes an illegitimate distortion, the processed Work should be categorized as inauthentic.

(a)

(b)

FIGURE 11.12

Two nearly identical images. Image (a) is the original. Image (b) has been JPEG compressed with a quality factor of 95%.

For many distortions, the correct group might seem obvious. For example, high-quality lossy compression, as illustrated in Figure 11.12, should probably be legitimate, in that it has essentially no perceptible effect. On the other hand, substantial editing, of the type illustrated in Figure 11.13, should probably be illegitimate, in that it can completely change the interpretation of the Work. Other distortions, however, are not as obvious. For example, consider low-quality lossy compression, as shown in Figure 11.14. The impact of this compression is perceptible, but few would consider it significant. Should it be legitimate?

FIGURE 11.13

Tampered version of the image in Figure 11.12(a).

FIGURE 11.14

Image after JPEG compression with a quality factor of 20.

The answer depends on the application. A basic rule of thumb is to consider the conclusions that might be drawn from Works when they are used as intended. Any distortions that will not change those conclusions should be legitimate. For example, medical images are generally used to help diagnose

illnesses and other disorders. If we disregard for the moment the legal issues regarding medical imaging, any distortion that will not affect a diagnosis can be considered legitimate.

Care must be taken when applying this rule of thumb. In some cases, a modified Work that appears to be an acceptable copy of the original might in fact lead to different conclusions. In a critical field such as medical imaging, therefore, the division of distortions into legitimate and illegitimate groups should be based on controlled studies, rather than on subjective judgment.

In many applications, the choice of which distortions to authorize is not only a technical decision but a legal one. In fact, if we were implementing an authentication system for medical images, we likely would be *primarily* concerned with the legal problems. This is even more true of other applications of authentication systems, such as authenticating evidence in criminal court cases. In these circumstances, the division of distortions into legitimate and illegitimate groups is a result of case law.

The set of legitimate distortions determined by case law is not always predictable. Sometimes, whether or not a distortion is legitimate depends as much on the relative skill of the lawyers arguing the first test case as it does on the technical merits of the argument. This can lead to some surprising inconsistencies. For instance, some distortions might be allowed if the processing took place in the spatial or temporal domain, but might not be allowed if the processing took place in the frequency domain, even if the processing is linear and the results are equivalent. There are also likely to be variations from country to country and state to state, and the set of legitimate distortions can change over time.

Given such inconsistent and changing requirements, an ideal selective authentication system would first identify the distortions that have been applied to a Work. Once these distortions have been identified, it is a straightforward matter to determine whether any of them are illegitimate. This is the aim of telltale watermarking, as discussed in Section 11.2.4. Of course, identifying those distortions a Work has undergone from an almost infinite set of possible distortions, many of which may not be known to the designer, is difficult, to say the least.

If the requirements are consistent and can be identified *a priori,* then an alternative system is possible in which we design a watermark that is only robust to the legitimate distortions. This arrangement does not attempt to identify the distortions a Work has undergone but simply outputs a single bit of information indicating whether the Work is considered authentic or inauthentic. This is the aim of semi-fragile watermarks and semi-fragile signatures, as discussed in Sections 11.2.2 and 11.2.3. Proving a watermark will survive legitimate distortions is straightforward. However, the converse problem of proving a watermark will not survive all illegitimate distortions is, once again, very difficult.

11.2.2 **Semi-Fragile Watermarks**

A *semi-fragile watermark* describes a watermark that is unaffected by legitimate distortions, but destroyed by illegitimate distortions. As such, it provides a mechanism for implementing selective authentication.

If the distinction between legitimate and illegitimate distortions is roughly based on perceptibility, creating a semi-fragile watermark is similar to creating a robust watermark. After all, the goal of robustness is to ensure that the watermark survives manipulations until the cover Work is so damaged that its value is lost. Beyond that point, we do not care whether a robust watermark survives. With a semi-fragile watermark, we still want to ensure that the watermark survives manipulation up to the point at which the Work's value is lost, but we also want to ensure that the watermark *does not* survive beyond that point. This often can be achieved by carefully tuning a robust watermark so that it is likely to be destroyed if the distortion exceeds a particular level. Several proposed semi-fragile systems are examples of this approach[5] [264, 346, 464].

The problem of designing a semi-fragile watermark becomes more difficult when the list of legitimate distortions is more specific, as is the case for the medical and legal applications discussed in Section 11.2.1. In these cases, we want our mark to survive certain distortions, and not survive others, even when the perceptual impact of the illegitimate distortions may be negligible. Thus, in these circumstances, the semi-fragile watermark must be designed with the specific legitimate distortions in mind.

There is a wide variety of distortions that might be considered legitimate, and each distortion might require its own specific type of semi-fragile watermark. However, for illustrative purposes, we will concentrate on one broad class of distortions: lossy compression by quantization in a transform domain.

As discussed in Chapter 9, many lossy compression systems involve converting a Work into some transform domain, such as the wavelet or block DCT domain, and then quantizing the coefficients to reduce their entropy. Coefficients are quantized either coarsely or finely, depending on how easy it is to perceive changes in them. How can we make a semi-fragile watermark that survives a certain amount of compression, applied by a specific compression algorithm, yet disappears after almost any other type of manipulation?

A possible answer lies in the following property of quantization. Let $x \diamond q$ be the result of quantizing x to an integral multiple of the quantization step size, q:

$$x \diamond q = q\left\lfloor \frac{x}{q} + 0.5 \right\rfloor. \tag{11.6}$$

[5] Most of the systems cited here also employ the idea of block-based authentication for localization. This is discussed in more detail in Section 11.3.

If a is a real-valued scalar quantity, and q_1 and q_2 are quantization step sizes, with $q_2 \leq q_1$, then

$$((a \diamond q_1) \diamond q_2) \diamond q_1 = a \diamond q_1. \tag{11.7}$$

In other words, if we quantize a to an even multiple of q_1, and subsequently quantize by q_2, we can undo the effect of the second quantization by again quantizing to a multiple of q_1, as long as $q_2 \leq q_1$. This can be exploited by having the watermark embedder and detector quantize transform coefficients by preselected step sizes [260]. As long as the quantization performed during compression uses smaller step sizes, the watermark should survive. This basic idea is employed to create the E_DCTQ/D_DCTQ authentication system, described in the following.

INVESTIGATION

Semi-Fragile Watermarking by Quantizing DCT Coefficients

The E_DCTQ/D_DCTQ image authentication system we present here uses a semi-fragile watermark designed to survive specific levels of JPEG compression. It is based on the system that Lin and Chang proposed in [260] (and extended to MPEG video in Lin and Chang [261]), but is simplified here to serve as a pure example of a semi-fragile watermark.[6]

In the JPEG image compression system, images are quantized in the block DCT domain. The quantization step size for each coefficient depends on its frequency. By default, the step sizes are obtained by multiplying a predefined quantization array (Table 11.1) by a given constant factor. The constant factor is usually a simple function of a "quality factor" between 0 and 100, which is entered by the user. Although it is possible for JPEG encoders to use more sophisticated algorithms (e.g., designing a unique quantization array for each image), the authentication system described here is targeted at the default JPEG behavior.

System 17: E_DCTQ/D_DCTQ

The E_DCTQ embedder embeds a pseudo-random bit pattern into an image. It takes two input parameters, other than the image to be watermarked: the seed for generating the pseudo-random bit pattern and controlling other pseudo-random operations, and a strength parameter, α, which specifies the level of JPEG compression the watermark should survive. This strength parameter is

[6] The example does not include the semi-fragile signature of Lin and Chang's system. This will be added in Section 11.2.3. It also does not include their system's localization feature (see Section 11.3) or restoration feature (see Section 11.4).

Table 11.1 Luminance quantization matrix used in JPEG. The upper left value (16) is the base quantization factor for the DC term of each 8×8 block. The lower right value (99) is the base quantization factor for the highest-frequency term. These base values are multiplied by a global quantization value to obtain the actual quantization factor used. (Note: This table is duplicated from Chapter 9.)

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

expressed as the largest multiplier that may be applied to the default matrix before an image should be considered invalid.

Four bits are embedded in the high-frequency DCT coefficients of each 8×8 block in the image. Let b be the value of one of the bits. The method for embedding this bit proceeds as follows:

1. Select seven coefficients, $\mathbf{C}[0], \mathbf{C}[1], \ldots, \mathbf{C}[6]$, from the set of 28 coefficients, shown shaded in Figure 11.15. The sets of coefficients selected for the 4 bits are disjoint, so that each coefficient is involved in only 1 bit. We ignore the low-frequency coefficients here because changing them is assumed to lead to unacceptably poor fidelity.

2. Divide each of the coefficients by its corresponding quantization factor, and round to the nearest integer. That is,

$$\mathbf{C_I}[i] = \left\lfloor \frac{\mathbf{C}[i]}{\alpha \mathbf{q}[i]} + 0.5 \right\rfloor, \tag{11.8}$$

   where $\mathbf{q}[i]$ is the value in Table 11.1 that corresponds to coefficient $\mathbf{C}[i]$.

3. Take the LSB of each of the resulting integers, $\mathbf{C_I}[0], \mathbf{C_I}[1], \ldots, \mathbf{C_I}[6]$, and exclusive-or them together to obtain the current bit value, b_e, represented by these coefficients.

4. If $b_e \neq b$, flip the LSB of one of the integers. The one to flip is the one that will cause the least fidelity impact. Let $\mathbf{C_{wI}}[0], \mathbf{C_{wI}}[1], \ldots, \mathbf{C_{wI}}[6]$

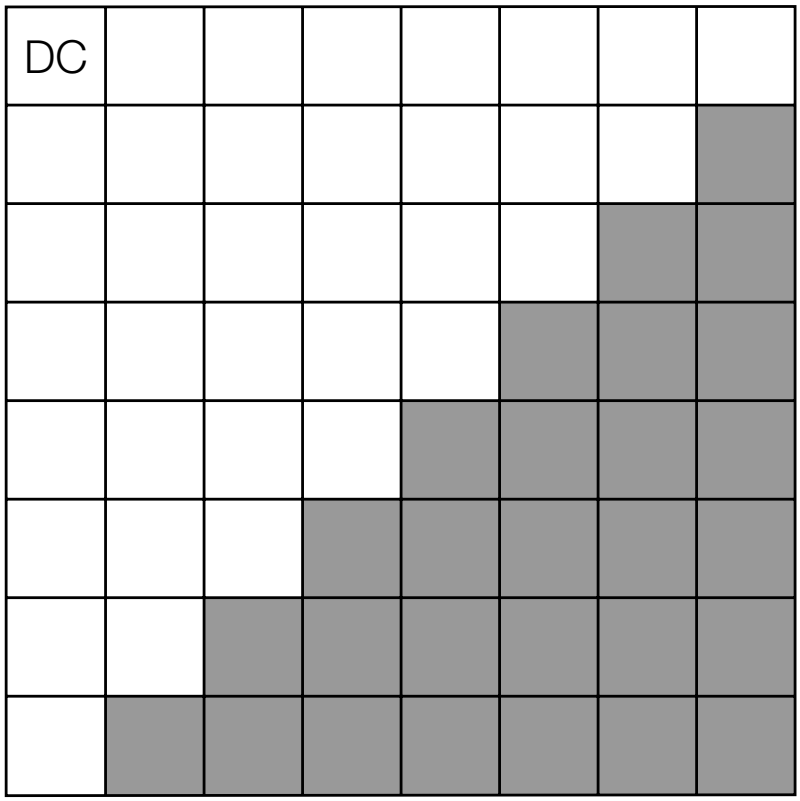


FIGURE 11.15

DCT coefficients used to embed a semi-fragile watermark in the E_DCTQ/D_DCTQ authentication system. The upper left element of this diagram corresponds to the DC term of each 8×8 block DCT. The shaded elements correspond to the terms used in the watermark.

denote the result. That is, $\mathbf{C_{wI}}[i] = \mathbf{C_I}[i]$ for all i unless $b_e \neq b$, in which case the LSB of one member of $\mathbf{C_{wI}}$ differs from that of the corresponding $\mathbf{C_I}$.

5. Multiply the members of $\mathbf{C_{wI}}$ by their corresponding quantization factors to obtain the watermarked versions of the DCT coefficients, $\mathbf{C_w}[0], \mathbf{C_w}[1], \ldots, \mathbf{C_w}[6]$:

$$\mathbf{C_w}[i] = \alpha \mathbf{q}[i]\mathbf{C_{wI}}[i]. \tag{11.9}$$

In theory, this algorithm should always succeed in embedding all bits. In practice, however, a few bits will be corrupted when the image is converted to the spatial domain, and each pixel is clipped and rounded to an 8-bit value. To compensate for this, we run the embedder on the image repeatedly, until all bits are correctly embedded.

The D_DCTQ detection algorithm takes as input an image to be authenticated and the same seed that was given to the embedder. It then simply performs the first three steps of the embedding algorithm to extract each bit, b_e. These are compared against the corresponding bits in the pseudo-random bit pattern, and the percentage of bits that match is compared against a threshold, τ_{match}. If this percentage is greater than the threshold, the image is declared authentic.

In theory, the threshold should be set at $\tau_{\text{match}} = 100\%$; that is, all bits should match. However, JPEG compression and decompression entails some clipping and round-off error, which sometimes corrupts embedded bits, even when the quantization multiplier is less than the embedding strength, α. Thus, we set $\tau_{\text{match}} = 85\%$.

Experiment

To test the performance of this system, watermarks were embedded in 2,000 images, with a strength of $\alpha = 0.3$. Each image was then subjected to some distortion and tested for authenticity with a threshold of $\tau_{\text{match}} = 85\%$. Two types of distortion were tested: DCT quantization and low-pass filtering.

The results for DCT quantization, which simulates JPEG compression, are shown in Figure 11.16. The horizontal axis indicates the multipliers for the quantization

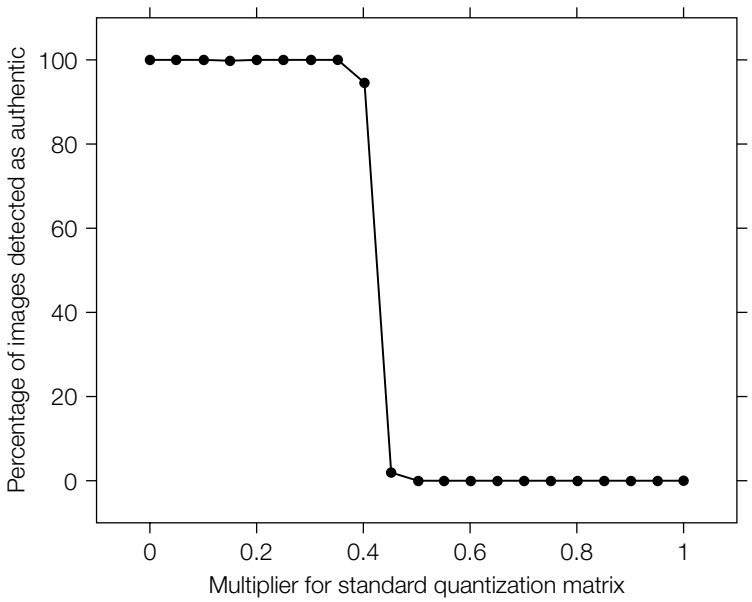


FIGURE 11.16

Results of quantization test of the E_DCTQ/D_DCTQ semi-fragile watermark authentication system.

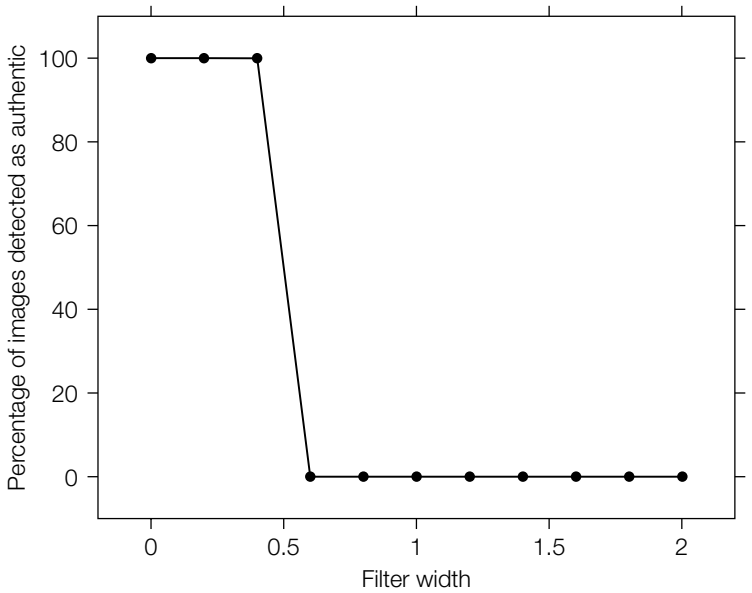


FIGURE 11.17

Results of low-pass filtering test of the E_DCTQ/D_DCTQ semi-fragile signature authentication system.

matrix. The vertical axis indicates the percentage of images that were detected as authentic. As expected, detection rates were largely unaffected when the multiplier was less than the strength parameter, and fell off rapidly at higher multipliers.

The results for low-pass filtering are shown in Figure 11.17. It is clear that even though the watermark survived fairly severe JPEG compression, even mild low-pass filtering quickly destroyed it. This performance is comparable with the performance of a white-noise watermark, tested in Chapter 9 (see Figure 9.9), which we judged to be very fragile. Thus, if the mark is detected, we can be reasonably sure that the image has not been low-pass filtered or JPEG compressed with a quantization multiplier much higher than 0.3.

11.2.3 Embedded, Semi-Fragile Signatures

Semi-fragile watermarks, like their fragile counterparts, are often not secure against malicious tampering. This is because they can succumb to copy attacks (see Section 10.3.3 of Chapter 10). In addition, semi-fragile watermarks are only able to authenticate those properties of an image they are embedded

within. For example, in the E_DCTQ/D_DCTQ system, the authentication watermark is only embedded in the high-frequency coefficients of the block DCT, because embedding in the low-frequency coefficients is too visible. However, this means that only changes in the high frequencies affect the watermark. If an illegitimate distortion changes the low frequencies, while leaving the high frequencies untouched, it will not affect the watermark, and the system will incorrectly report the image as authentic.

These problems can be addressed by identifying features of a Work that are invariant to legitimate distortions, but not to illegitimate distortions, and using them to construct a signature [361, 391]. Because the signature is unaffected by legitimate distortions, but changed by others, we refer to it as a *semi-fragile signature*.

Like the cryptographic signatures used for exact authentication (Section 11.1.2), a semi-fragile signature can be embedded as a watermark. However, in this case, the watermark cannot be fragile, because it must be able to survive any legitimate distortion. An appropriately designed semi-fragile watermark can be useful here, in that such a watermark might complement the signature. That is, although both the signature and the watermark are designed to survive legitimate distortions, they might be fragile against different sets of illegitimate distortions.

There are at least two advantages of using semi-fragile signatures. First, in such a system, each Work has a different watermark embedded. This means that an adversary cannot make a forgery appear authentic by performing a simple copy attack.

Second, the signature can be based on properties of the Work that we cannot change without causing unacceptable fidelity problems. For example, we can base a signature on the perceptually significant components of a Work [38, 279] in some transform domain, such as the low-frequency coefficients of the block DCT. We can then embed this signature in more easily alterable coefficients, such as the high-frequency coefficients of the block DCT. Of course, the embedded signature must still survive legitimate distortions. This arrangement is illustrated with the E_SFSIG/D_SFSIG authentication system.

INVESTIGATION

Semi-Fragile Signatures Embedded with Semi-Fragile Watermarks

We consider a system that extracts a signature from the low-frequency terms of the block DCTs of an image and embeds it in the high-frequency terms using a semi-fragile watermark. Both the semi-fragile signatures and the semi-fragile watermarks are designed to survive JPEG compression up to a given quantization multiplier, but are altered by most any other process. The method of extracting these signatures is taken from Lin and Chang [260].

System 18: E_SFSIG/D_SFSIG

In this system, the embedder, E_SFSIG, extracts a semi-fragile signature from each image and then uses E_DCTQ to embed it. To extract signatures, we take advantage of a property of quantization. That is, if two values are quantized by the same step size, the resulting values maintain the same relation to one another. That is, if $a > b$, then $a \diamond q \geq b \diamond q$, where $\diamond$ is the quantization operator defined in Equation 11.6.

A signature for the image is extracted as follows:

1. Convert the image into the block DCT domain.
2. Group the blocks of the image into pseudo-random pairs, according to a given seed.
3. In each pair of blocks, compare n corresponding low-frequency coefficients to obtain n bits of the binary signature. That is, we compute each of the n bits as

$$\text{bit} = \begin{cases} 0 & \text{if } \mathbf{C}[i, j, k_0] < \mathbf{C}[i, j, k_1] \\ 1 & \text{if } \mathbf{C}[i, j, k_0] \geq \mathbf{C}[i, j, k_1] \end{cases}, \tag{11.10}$$

where i, j are the coordinates of a low-frequency coefficient, $\mathbf{C}[i, j, k]$ is the i, jth coefficient in the kth block of the image, and k_0 and k_1 are the indices of two blocks that have been grouped. The shaded elements of Figure 11.18 indicate the set of coefficients used to compute signatures in our experiments.

During standard JPEG compression, every block is quantized by the same quantization table. Therefore, ideally JPEG should not change our signatures. The only changes that should occur happen when corresponding coefficients in two paired blocks are quantized to the same value (often 0). If $\mathbf{C}[i, j, k_0]$ was originally less than $\mathbf{C}[i, j, k_1]$, this quantization causes the signature bit to change from a 0 to a 1. Changes can also occur as a result of clipping and rounding after JPEG quantization, and this is particularly a problem when two paired coefficients are close to equal.

The E_SFSIG embedder computes a set of signature bits and embeds them into the high-frequency coefficients of the block DCT in a pseudo-random order, according to the same method used in E_DCTQ. The D_SFSIG detector extracts the embedded bits and compares them to the signature it computes from the image. During this comparison it ignores any signature bits computed from pairs of coefficients that are close to equal (wherever $|\mathbf{C}[i, j, k_0] - \mathbf{C}[i, j, k_1]| < \varepsilon$), because these might have been flipped during JPEG quantization. It then computes the percentage of the remaining signature bits (those extracted from unequal coefficient pairs) that match the embedded bits. If this percentage is over a given threshold, τ_{match}, it reports that the image is authentic.

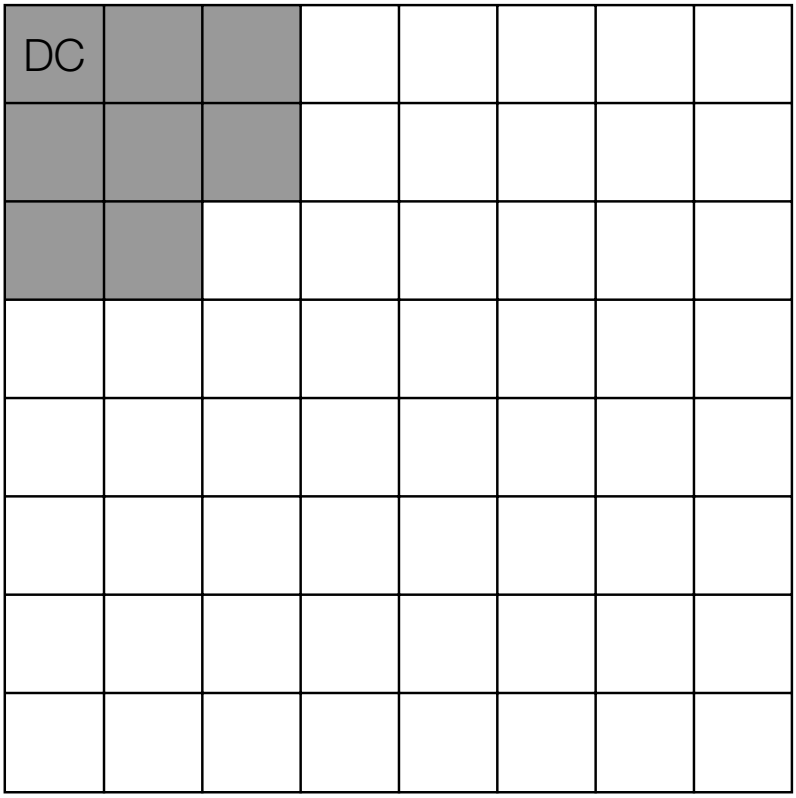


FIGURE 11.18

DCT coefficients used to compute semi-robust signatures in the E_SFSIG/D_SFSIG authentication system. The upper left element of this diagram corresponds to the DC term of each 8×8 block DCT. The shaded elements correspond to the terms used in the signature.

Experiment

To test the performance of this system, watermarks were embedded in 2,000 images, with a strength of $\alpha = 0.3$. Each image was then subjected to some distortion and tested for authenticity with $\varepsilon = 4$ and a threshold of $\tau_{\text{match}} = 85\%$. Three types of processing were tested: DCT quantization, low-pass filtering, and additive low-frequency noise.

Figure 11.19 shows the results of quantization, and Figure 11.20 shows the results of low-pass filtering. Because the semi-fragile signature is relatively unaffected by these two processes (except as a result of round-off and clipping), the authenticity results were very similar to those from the E_DCTQ/D_DCTQ authentication system (Figures 11.16 and 11.17).

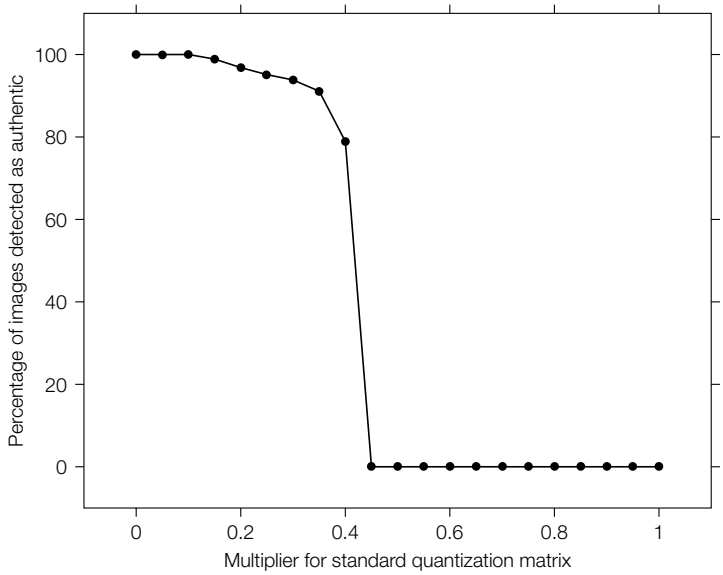


FIGURE 11.19

Results of quantization test of the E_SFSIG/D_SFSIG authentication system.

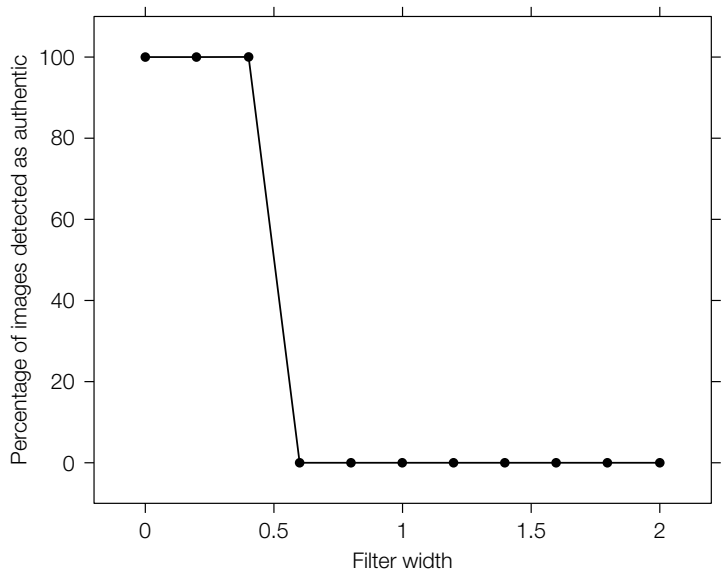


FIGURE 11.20

Results of low-pass filtering test of the E_SFSIG/D_SFSIG authentication system.

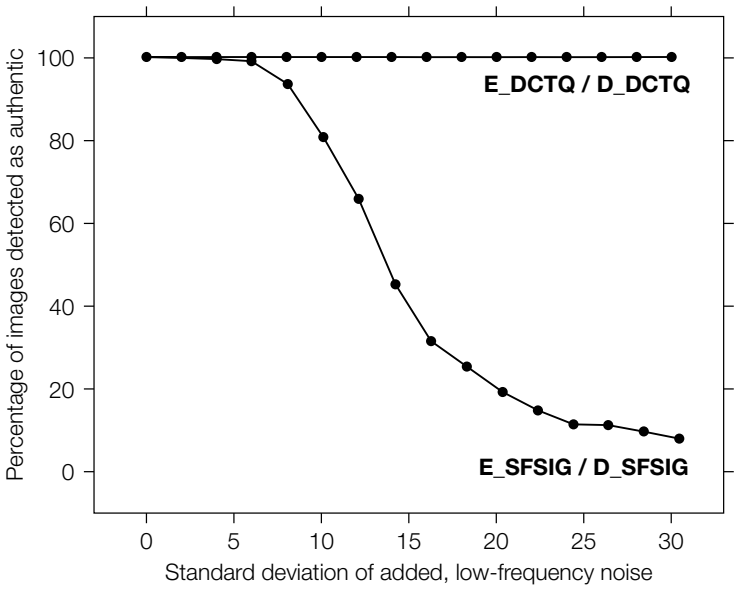


FIGURE 11.21

Results of low-frequency noise test of the E_DCTQ/D_DCTQ and E_SFSIG/D_SFSIG authentication systems.

The two systems, E_DCTQ/D_DCTQ and E_SFSIG/D_SFSIG, are expected to behave differently when the distortion affects the low-frequency components of the image. In this case, the semi-fragile signature should become invalid. Watermarks were embedded in 2,000 images using both the E_DCTQ and the E_SFSIG embedder, and each of the 4,000 watermarked images was corrupted by the addition of low-frequency noise. This noise has little effect on the high-frequency terms of the block DCTs. Therefore, the semi-fragile watermark should survive. However, in the E_SFSIG/D_SFSIG system the low-frequency noise will change the semi-fragile signature, so that it will no longer match the embedded signature.

The results are plotted in Figure 11.21. Although the E_DCTQ/D_DCTQ system reports a large percentage of images as authentic, even at very high levels of noise, the E_SFSIG/D_SFSIG system identifies a decreasing number of images as authentic as the amount of noise increases.

11.2.4 **Telltale Watermarks**

The two semi-fragile approaches previously described are appropriate for applications in which there is a clear distinction between legitimate and illegitimate distortions. However, in applications for which this distinction might change

over time, or differ from place to place, we would like to obtain information about *how* a Work has been corrupted, rather than simply *whether* it has been corrupted. One possible way of investigating how a Work has been corrupted is to examine how a known, embedded watermark has been corrupted. This type of watermark has been referred to as a *telltale watermark*.

In [242, 464] it has been suggested that a watermark be embedded in the wavelet domain, using a form of quantization marking similar to that in the E_DCTQ/D_DCTQ system. The authenticator then examines various subsets of the wavelet decomposition, finding the number of embedded bits in each band that do not match the original watermark. This gives an idea of how badly each subband has been corrupted. Because different distortions corrupt different subbands of wavelet coefficients, it is possible to hypothesize how the Work has been distorted. For example, if all low-frequency coefficients are substantially corrupted but high-frequency coefficients are fairly intact, we can conclude that a high-pass filter has been applied to the Work. If all frequencies in a narrow spatial area have been equally corrupted, we might conclude that the area has been substantially edited.

The most general forms of telltale watermarks would distinguish between a wide variety of different distortions. Research into such general schemes is still in its infancy. However, there has been substantial research into the narrower problem of using watermarks to determine *where* a Work has been altered. Although this can be thought of as a special case of telltale watermarking, it has received enough attention to warrant an independent discussion.

11.3 LOCALIZATION

Many authentication methods based on watermarking have the ability to identify times or regions of the Work that have been corrupted, while verifying that the remainder of the Work has not been changed. This capability is referred to as *localization*.

Localization is useful because knowledge of when or where a Work has been altered can be used to infer (1) the motive for tampering, (2) possible candidate adversaries, and (3) whether the alteration is legitimate. For example, consider a photograph of a vehicle traveling along a highway. If our authenticator simply states that the image has been modified, the tampered image is useless. However, if the authenticator also indicated that the modification only occurred within the region of the vehicle's license plate, then the photograph is still very useful in determining the make, model, and color of the car.

This section briefly discusses two closely related approaches to localization. The first, *block-wise authentication,* divides a Work into contiguous blocks and embeds an authentication watermark into each block independently. The second, *sample-wise authentication,* is an extreme case of block-wise authentication in which each block is reduced to the size of one sample. The section

then goes on to discuss security issues that must be considered when using a localized authentication method.

11.3.1 Block-Wise Content Authentication

Most localized authentication methods rely on some form of *block-wise authentication,* in which the Work is divided into a number of disjoint temporal or spatial regions, each of which is authenticated separately. If part of the Work is modified, only the affected regions fail to authenticate.

INVESTIGATION

Block-Wise Authentication

The E_DCTQ/D_DCTQ authentication system can be easily modified to perform localization, as modification of any given 8×8 block in an image affects only the bits embedded in that block. This is done by modifying the D_DCTQ detector to measure the number of correctly embedded bits in each independent block, and output a map of which blocks are valid and which are not. The change to the D_DCTQ detector is in the reporting of results rather than in the detection algorithm, therefore we will not introduce a new system name here.

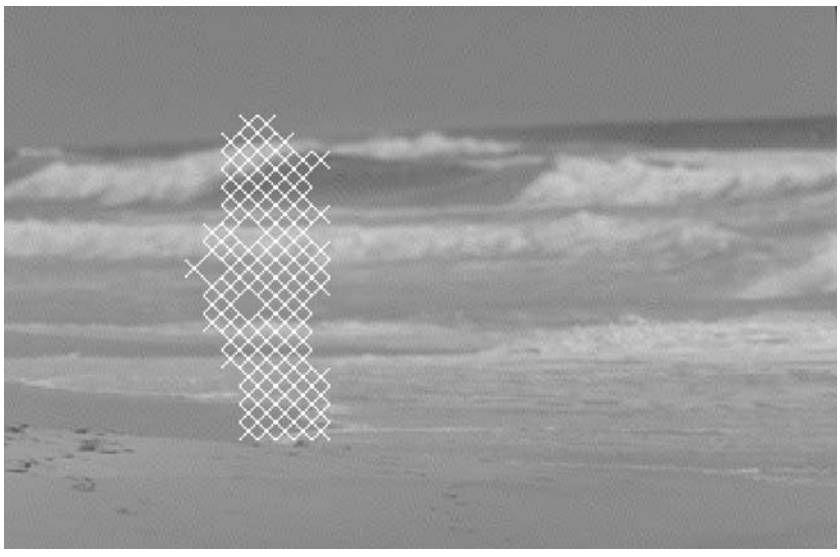


FIGURE 11.22

Localized authentication using the E_DCTQ/D_DCTQ system.

This investigation is intended to simply illustrate the localization functionality gained by examining each block independently. We show this by applying the localized authentication system to an image and modifying the image prior to presentation to the detector.

First, we embedded an authentication mark into the image in Figure 11.12(a), using E_DCTQ. Then, the watermark image was tampered with, and the woman was removed from the image by copying all the altered pixels from Figure 11.13 into the corresponding locations in the watermarked image. Finally, we used the D_DCTQ algorithm to label the inauthentic blocks. Figure 11.22 shows the results. Each block that was labeled inauthentic by the D_DCTQ detector is marked with an x.

11.3.2 Sample-Wise Content Authentication

The spatial acuity with which a block-based authentication system localizes tampering depends on the block size. Because smaller block sizes lead to more precise localization, it might be desirable to reduce the blocks to single samples, resulting in *sample-wise authentication*. However, as we shall see, there are potential security risks associated with small block sizes, and these risks are greatest in sample-wise authentication. To illustrate the security risks, we first present an example sample-wise authentication system.

INVESTIGATION

Pixel-Wise Image Authentication

The pixel-wise image authentication system presented here was first described by Yeung and Mintzer in [463] and is probably one of the most widely studied systems for localized authentication. This means that many of the attacks we will be describing have been applied to it.

System 19: E_PXL/D_PXL

The extraction process employs a pseudo-random mapping from pixel intensities into binary values. For example, intensities 0 through 3 might map to a bit value of 1, intensities 4 and 5 to a bit value of 0, intensities 6 through 9 to a bit value of 1, and so on. This mapping is determined by a table constructed by a pseudo-random number (PN) generator. The seed to the PN generator serves as a secret key. Each pixel in the image thus holds 1 bit of watermark information. The bit patterns used in Yeung and Mintzer [463] are tiled, binary images, usually in the form of a logo, as illustrated in Figure 11.23.

Note that an LSB watermark is a special case of this system, in which the mapping from intensities to bit values simply alternates between 1 and 0; that is,

even-valued intensities map to 0 and odd-valued intensities map to 1. The use of a pseudo-random mapping, however, makes copy attacks more difficult.

The E_PXL embedder compares the extracted mark to the reference mark pixel by pixel. At pixel locations that do not match, the image values are replaced with the closest match from the mapping table. Thus, assuming the previously described intensity mapping, if the original pixel intensity is 3 and the corresponding watermark value is 0 at this location, then the embedder would alter the pixel value to 4.

Of course, when the watermark embedder alters pixel intensities, visible artifacts can be introduced. To reduce the risk of artifacts, we employ a common halftoning technique known as *error diffusion* [219]: As each pixel is modified, a fraction of the error introduced in its value is added to each of the neighboring pixels that have not yet been modified. Because the embedder proceeds from top to bottom, left to right, this means we distribute the errors down and to the right. Figure 11.24 shows the weights used for this distribution.

The D_PXL detector simply extracts the mark using the pseudo-random mapping table, thereby generating a binary pattern. If the image has not been modified, this extracted pattern exactly matches the reference pattern. However, if a region has been modified, these alterations will likely show up as noise in the binary pattern.

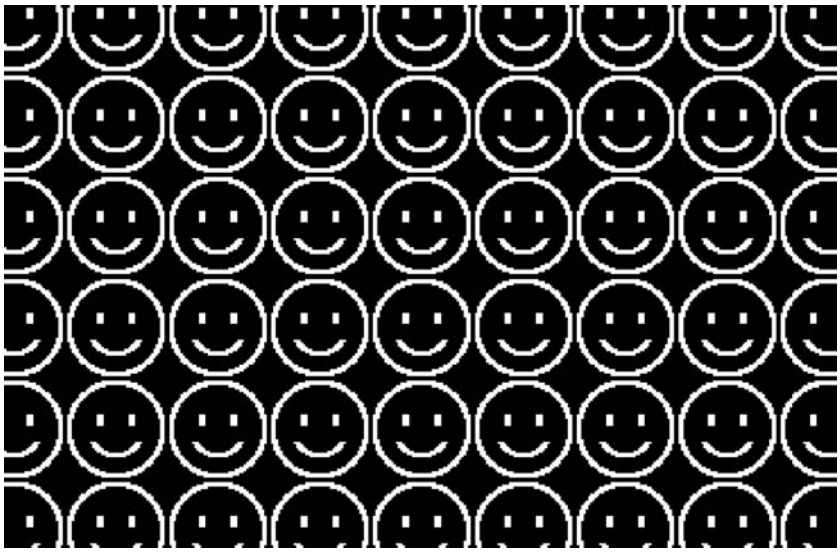


FIGURE 11.23

A tiled, binary pattern that can be used as a watermark in the E_PXL/D_PXL authentication system.

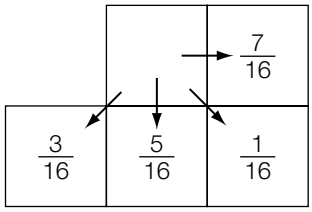


FIGURE 11.24

Weights used for error diffusion in the E_PXL/D_PXL authentication system (from Kang [219]).

Experiment

To illustrate the performance of this algorithm, we performed an experiment similar to that in Figure 11.22. The watermark pattern of Figure 11.23 was embedded into the image in Figure 11.12(a), using the E_PXL embedder. Pixels from Figure 11.13 were then pasted into the watermarked image to remove the woman from the scene. Figure 11.25 shows the pattern extracted by the D_PXL detector. Most of this pattern matches the original watermark pattern, but the area containing the woman is corrupted.

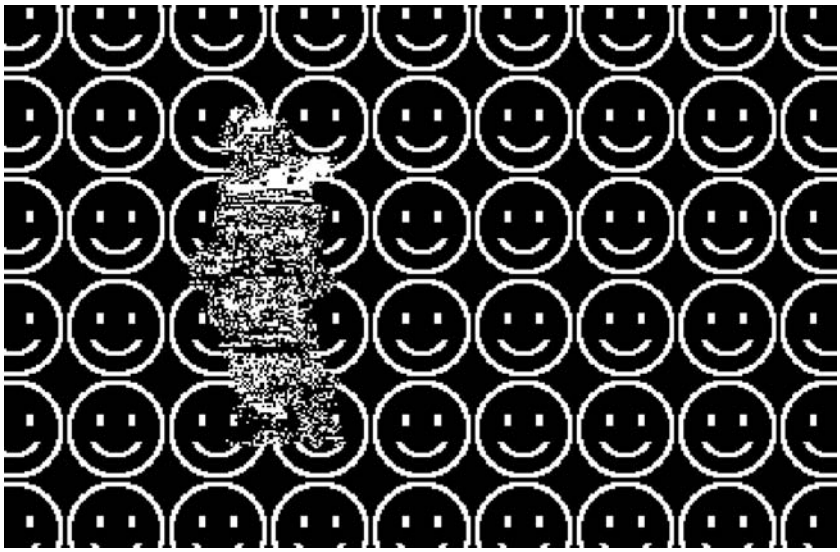


FIGURE 11.25

Pattern extracted from a tampered image by the E_PXL/D_PXL authentication system.

11.3.3 Security Risks with Localization

There are a number of security risks associated with localized authentication systems. Although the risks discussed here can be countered with simple modifications, or proper use of the systems, it is important to be aware of them.

We are concerned with forgery attacks in which an adversary wishes to embed a valid watermark into either a modified or counterfeit Work. Two basic categories of attack, each of which requires different resources, will be examined. In *search* attacks, the adversary is assumed to have a black-box detector that can determine whether a Work is authentic or not. In *collage* attacks, the adversary is assumed to have two or more Works embedded with the same watermark key.

Search Attacks

Let's consider the situation in which everyone, including potential adversaries, has access to a watermark detector. This situation might arise, for example, if Works are being distributed to the general public over the Internet, and we want to use an authentication system to guarantee that each Work is delivered without corruption.

Under these circumstances, the adversary can, in theory, use the detector to thwart any authentication system (regardless of whether it is localized). To do so requires a brute-force search. To embed a forged watermark into a Work, an adversary can enter slightly modified versions of the Work into the detector until one is found that the detector reports as authentic.

In practice, of course, this search would usually be prohibitive. However, with a block-wise or sample-wise authentication system, the search space can be considerably smaller. The adversary can perform a separate, independent search on each block. If the block size is small enough—especially if it is a single sample—the search becomes feasible.

Such attacks can be countered by choosing a larger block size. The E_PXL/D_PXL system is clearly susceptible, in that the search would require *at most* 256 iterations per pixel (one for each possible intensity). The E_DCTQ/D_DCTQ system, with its 64-pixel blocks, is much safer.

Collage Attacks

The second category of attacks relies on having access to one or more authentic watermarked Works. By examining these Works, an adversary can come up with sets of blocks that are all authentic and construct a counterfeit Work from them like a "collage."

Holliman and Memon [195] described an attack applicable to block-wise watermarks. They considered the case in which a cryptographic signature is embedded in each block (see Section 11.1.2), and the signature depends only on the content of the block itself. This is referred to as *block-wise independence*. Consider what happens in such a system when two blocks of a

watermarked Work are interchanged, thereby changing the Work as a whole. Because each block contains a self-authenticating watermark, and because each block remains unaltered, the work is deemed authentic. Thus, even if all blocks are scrambled into a random order, the system will regard the entire Work as authentic.

By exploiting this weakness of block-wise independent systems, it is possible to create a completely new Work that is assembled from the set of independent, authentic blocks. Suppose an adversary has a number of Works available, all watermarked using the same key. This can be viewed as a large database of authentic blocks from which a new image can be built. To forge a watermark in an unwatermarked Work, the adversary divides the Work into blocks and replaces each block with the most similar block from the database. With a large enough database of watermarked Works, the results may be quite good. Figure 11.26 shows an image constructed out of 8 × 8 blocks from 500 images. The fidelity of the forgery is surprisingly high, although some block artifacts are present. However, the number of such artifacts will diminish as the size of the database increases.

This attack only applies to block-wise independent systems in which the mark embedded in a block is independent of the block's location. However, systems such as E_DCTQ/D_DCTQ and E_PXL/D_PXL, which embed different information into different locations, also can be attacked in a similar fashion. An example of this attack on a sample-wise authentication system, presented in Fridrich *et al.* [153], is described in the following.

FIGURE 11.26

Image constructed by assembling 8 × 8 blocks from 500 different images.

The idea is very simple. For each sample of an unwatermarked image, look through the set of watermarked Works to find the one with the closest sample value in the same location. Then replace the sample in the unwatermarked Work with the one that was found, and use error diffusion to compensate for any fidelity impact.

The pixel-wise forgery attack of Fridrich *et al.* [153] is implemented using a database of 100 images authenticated with the E_PXL embedder. The goal is to use these images to forge a new image that will be considered authentic by the D_PXL detector.

Figure 11.27 shows the result of this attack. In this case, the forgery is nearly perfect and yet only 100 images were needed, significantly less than for the block-based method illustrated in Figure 11.26. The D_PXL detector reports this image as authentic. However, this attack would not fool the D_DCTQ detector because neither the relationships between the pixels nor the relationships between the DCT coefficients within a block are maintained.

If the adversary obtains two or more Works known to contain the same watermark, it is sometimes possible to learn enough about that watermark to embed it in another Work. To illustrate this threat, consider the situation in which blocks at the same locations in different Works contain the same watermark information. Using this fact, the adversary can identify sets of blocks that can legitimately appear in each location. Armed with this knowledge, the adversary can forge a watermark in a new Work by replacing each of its blocks with the most similar block in the corresponding set.

FIGURE 11.27

Image constructed by assembling pixels from 100 different watermarked images.

A more sophisticated form of attack on sample-wise authentication systems was described in Fridrich *et al.* [153]. Consider two images, both of which have had the same binary watermark embedded using the E_PXL embedder. If the upper left pixel in one image is 0, and the corresponding pixel in the other image is 2, then we know that both 0 and 2 map into the same bit value of the watermark. If at another location the first image has a 2 and the second image has a 25, we know that 0, 2, and 25 all map to the same bit value. Using this knowledge, we can build up a nearly complete picture of the pseudo-random mapping function at the heart of the E_PXL/D_PXL system.

With two images, the algorithm proceeds as follows:

1. Initialize a collection of sets of intensity values, $\mathcal{S}$, to contain 256 sets, each containing one intensity value. That is,

$$\mathcal{S} \leftarrow \{\{0\}, \{1\}, \{2\}, \ldots, \{255\}\}. \tag{11.11}$$

2. For each pixel location, identify the two sets in $\mathcal{S}$ that contain that location's intensities in the two images. Then combine those two sets. For example, if the first pixel in one image is intensity 0, and in the other image is intensity 2, the two sets, $\{0\}$ and $\{2\}$, would be combined, thus:

$$\mathcal{S} \leftarrow \{\{0, 2\}, \{1\}, \{3\}, \ldots, \{255\}\}. \tag{11.12}$$

   If the next location contained 2 in one image and 25 in the other, $\{0, 2\}$ and $\{25\}$ then would be combined. And so on. After visiting every pixel location, $\mathcal{S}$ will contain only a few sets (perhaps just two), and we know that all of the pixel values in each set map into the same bit value in the watermark pattern.

3. Having found $\mathcal{S}$, we can now embed a forged watermark in an unwatermarked Work. At each pixel location, identify the set in $\mathcal{S}$ that contains the value in one of the *watermarked* Works. Then replace the pixel in the *unwatermarked* Work with the closest value in that set. Use error diffusion to compensate for the damage to fidelity.

In our example, one of the Works has a 0 in the first pixel location. Let us say the set in $\mathcal{S}$ that contains 0 is $\{0, 2, 27, 34, 89, 126, 134, 158, 230\}$. We know that all of these intensities map into the same watermark bit value. We also know that whatever it is, that value is the correct value for the first pixel location. Suppose the first pixel in the unwatermarked Work has a value of 136. We would change this to the closest value in the set, 134, and distribute the error to neighboring pixels.

In principle, this same attack could be applied to larger block sizes, but it quickly becomes impractical as the number of possible blocks increases. Even with the 8×8 blocks of the E_DCTQ/D_DCTQ system, the size of $\mathcal{S}$ is prohibitive.

The surest way to counter these types of attacks is to use a different key for watermarking every Work. However, such an approach is not always feasible, in that a given Work can only be authenticated if the correct key is available. The keys would need to be either known to the users or stored as associated data (e.g., in a header field of each Work). The application might prohibit both of these solutions.

A more practical approach, suggested in Holliman and Memon [195], is to make the signature in each block depend on some of the surrounding data, as well as the data within the block itself. This introduces some ambiguity to the localization, because a change in one block can change the signature that should be embedded in its neighbors. However, it dramatically complicates the adversary's attempt to build a Work out of watermarked blocks, as each block must match up properly with the neighboring blocks.

11.4 RESTORATION

We have seen that it is possible to determine if a Work has been altered, where a Work has been altered, and even how a Work has been altered. This naturally leads to an examination of whether an altered Work can be restored.

There are two basic restoration strategies: *exact restoration* and *approximate restoration*. As the names suggest, exact restoration seeks to restore the Work to its original state (i.e., the goal is a perfect copy with not a single bit in error). This is a very well-studied problem in communications and is discussed in Section 11.4.1.

Approximate restoration is a more recent concept that seeks to restore a Work while accepting that there will be differences between the restored and original Works. However, the restored Work is still valuable if these differences are not significant. Sections 11.4.2 and 11.4.3 describe two approaches to approximate restoration. In the first, additional information is embedded in the Work to assist in the restoration. In the latter, the watermark is first analyzed to determine how the Work has been distorted, and this information is then used to invert the distortion. Clearly, the latter process is only applicable if the distortion is invertible.

11.4.1 Embedded Redundancy

It is well known that error detection and correction codes allow changes in data to be detected and, in many cases, corrected [163]. Error correction codes (ECCs) are widely used in communication and data storage to maintain the integrity of digital data. ECCs are similar to digital signatures in that the code bits are typically appended to the data. However, there are also important differences between ECCs and signatures.

The purpose of a digital signature is, of course, to verify that the data has not changed. The use of a one-way hash function makes the probability of an undetected change exceedingly small. In contrast, ECCs usually assume a maximum number of bit changes. If this number is exceeded, it is possible for errors to go undetected. However, with careful design, the likelihood of missed errors can be low. It is possible to use a digital signature in conjuction with an ECC.

The size of an ECC is usually very much larger than a digital signature. In fact, an ECC can represent a significant fraction of transmitted bits. The size of the ECC determines both the maximum number of bit changes that can be detected and the maximum number of bits that can be corrected. These two numbers are sometimes different, since some ECCs are able to detect many more changes than they can correct.

A Work can be considered simply as a collection of bits, and a variety of different ECCs can be applied (e.g., Hamming codes, turbo codes, trellis codes, etc.). Once more, this metadata can be represented using a watermark. For example, a Reed Solomon ECC can be used to generate parity bytes for each row and column of an image [252, 253]. These parity bytes can be embedded as a watermark in the two LSB planes of the image.[7] It is reported in [252, 253] that for a 229×229 image up to 13 bytes in a single row or column can be corrected. Even if the errors cannot be corrected, they can be localized, because parity bytes are calculated for each row and column.

A variant to this method exploits the assumption that errors often come as bursts [252, 253]. This is especially true if a localized region of an image has been modified or cropped. To increase the resistance to burst errors, the locations of the pixels in the image are randomized prior to calculation of the ECCs. This randomization is a function of a watermark key.

Clearly, if we want to restore a Work to its original state, a very significant cost must be incurred to store the ECCs. If this cost is too high, or the resources are simply unavailable, then approximate restoration techniques may be a good compromise. Two such approaches are discussed next.

11.4.2 Self-Embedding

Several authors [138, 259] have proposed "self-embedding," in which a highly compressed version of the Work is embedded in the Work itself. Thus, if portions of the watermarked Work are removed or destroyed, these modified regions can be replaced with their corresponding low-resolution versions.

To illustrate self-embedding, we briefly describe the fundamental ideas behind an algorithm proposed by Fridrich and Goljan [138]. Their method takes

[7] Thus, if 8 bits are used to represent each pixel intensity value, the ECC consumes 25% (2/8) of the available data—very much more than a digital signature.

an original image Work and extracts a JPEG compressed version at about a 50% quality factor. This low-resolution image requires only 1 bit per pixel and thus can be inserted in the LSB plane of the Work. However, rather than insert each compressed DCT block in the LSB of its corresponding spatial block, the binary sequence is first encrypted and then inserted in the LSB plane of a block that is at least a minimum distance away and in a randomly chosen direction. The authors suggest a minimum distance of 3/10 the image size. The random mapping is generated using a key that must be known to both the embedder and detector. Storing the low-resolution version of a block some distance away from the corresponding block allows this block to be restored even if it has been completely deleted. Of course, a higher-quality reconstruction is always possible if more bits are allocated to the storage of the low-resolution Work. The method is severely affected by any modifications to the encoding plane. However, more robust embedding methods can be used, as described in Fridrich and Goljan [138].

Although this example uses a comparable number of bits to the ECC method of Lee and Won [252], its "correction" capability is very much greater. It can essentially correct the entire image, albeit at a lower resolution.

11.4.3 Blind Restoration

An alternative model for the approximate correction of errors is based on *blind restoration*. Blind restoration [240] attempts to first determine what distortions a Work has undergone, and then to invert these distortions to restore the Work to its original state. Such a process is only appropriate if the distortion is invertible. Thus, blind restoration is not useful against, for example, clipping.

In Section 11.2.4 we described how telltale watermarks can be used to determine *how* a Work has been distorted. We also described one such method, proposed in Kundur and Hatzinakos [242]. This work on telltale watermarks has been extended to support image restoration [241] under the assumption that the image degradation can be modeled by localized linear blurring. This information is then used to approximately invert the blur distortion.

The fact that the watermark undergoes the same distortions as the cover Work suggests that a carefully designed watermark might be very useful in estimating the form of distortion a Work has undergone. Once this distortion is known, it is relatively straightforward to invert the process, assuming, of course, that the distortion is invertible. A combination of blind restoration and self-embedding may also be appropriate. In principle, blind restoration might allow a lower-resolution Work to be embedded. This is because at least some of the distortion may be invertible. In addition, where clipping or other noninvertible distortions have been applied, the self-embedded information allows for a low-resolution restoration. The complementary nature of the two methods is an interesting research topic.

11.5 SUMMARY

This chapter covered several issues related to the use of watermarks for authenticating content. The following main points were made.

- Several questions may be asked about the authenticity of a Work, including:
  - Has the Work been altered in any way whatsoever?
  - Has the Work been *significantly* altered?
  - What parts of the Work have been altered?
  - Can an altered Work be restored?
- Although other techniques for answering these questions exist, watermarks may be useful because they do not require auxiliary data and they undergo the same transformations as their cover Works. However, whether these advantages outweigh their disadvantages remains to be seen.
- An *exact authentication* system seeks to verify that a Work has not been changed *at all*. We discussed two basic methods for doing this:
  - *Fragile watermarks* are designed to become undetectable with the slightest change in their cover Works.
  - *Embedded signatures* are simply cryptographic signatures (as discussed in Chapter 10) embedded as watermarks.
- It is sometimes possible to create an *erasable watermark,* which, after being detected, can be removed to obtain an exact copy of the original. However, this type of watermark must be designed with careful consideration of the application to avoid unacceptable embedding effectiveness or false positive probability. Three particular erasable watermarking methods were discussed in detail based on:
  - Lossless compressibility.
  - Difference expansion.
  - Histogram modification.
- A *selective authentication* system seeks to verify that a Work has not been modified by any of a predefined set of *illegitimate distortions,* while allowing modification by *legitimate distortions*. We discussed three basic approaches for doing this:
  - *Semi-fragile watermarks* are designed to survive legitimate distortions but be destroyed by illegitimate distortions.

  - *Semi-fragile signatures* are signatures computed from properties of the content that are unchanged by legitimate distortions. These can be embedded in content using robust or semi-fragile watermarks.
  - *Telltale watermarks* are designed to be examined in detail after a Work is modified. By determining how the watermark has changed, we can infer how the Work has been distorted and make a subsequent determination as to whether or not the distortion was legitimate.
- *Localization* refers to the ability of an authentication system to identify which portions of a Work are authentic and which are corrupted. This is usually done by dividing the Work into parts and authenticating each part separately. However, there are some security concerns with the most straightforward approaches.
- *Restoration* refers to the ability of a system to restore portions of a Work that have been corrupted. We discussed three basic approaches:
  - *Embedded redundancy* embeds error correction bits in a Work as a watermark.
  - *Self-embedding* embeds a low-quality copy of a Work in a robust watermark.
  - *Blind restoration* uses a telltale watermark to identify distortions that have been applied to a Work, and then attempts to invert those distortions.

This page intentionally left blank

CHAPTER 12

Steganography

Steganography is the act of covert communications, which means that only the sender, Alice, and receiver, Bob, are aware of the secret communication. To accomplish this, the secret message is hidden within benign-looking communications known as covertexts or cover Works. To an adversary, Eve, it is clear that Alice and Bob are communicating, but the combined covertext and hidden message, referred to as a stegotext or stego Work, appears to be innocuous (i.e., Eve is unaware that the innocuous content hides a message).

The main requirement of steganography is *undetectability*, which, loosely defined, means that no algorithm exists that can determine whether a Work contains a hidden message. Steganalysis is the process of detection of steganographic communications. And since steganography and steganalysis are closely intertwined, some of the following discussion will oscillate between one and the other.

Steganography and watermarking are both forms of data hiding and share some common foundations. Nevertheless, it is worth reiterating the goals of these two data-hiding applications in order to highlight the key differences.

In Chapter 1, we defined watermarking as the practice of imperceptibly altering a Work to embed a message about that Work. This message might, for example, be a digest or hash of the Work, ownership information, a unique identifier of the sender or the consumer, or digital rights management information. However, whatever the message, the number of bits needed to be embedded is relatively small—typically 8 bits and almost never more than 128. The fact that a Work contains a watermark is often widely known. Indeed, in many applications, knowledge of the use of watermarking is actively promoted to deter and discourage illegal use of the cover Work. And it is the cover Work that is of principal value, thereby requiring that the watermark be imperceptible.

> *We define steganography as the practice of undetectably altering a Work to embed a message.*

Comparing this definition with that of watermarking, we note that we no longer require imperceptibility, but rather undetectability. Thus, in theory, the change

may be perceptible, as for example, when an image of a man in a blue suit is altered to one of a man in a black suit. Perceptibility is permitted because the cover Work has no intrinsic value. And a perceptible change may still be undetectable (i.e., no algorithm exists that is able to determine whether a Work contains a hidden message), since the adversary does not have access to the original, unmodified cover Work. Since the message no longer needs to be "about that Work," but rather is arbitrary, the desired length of the message may be very much longer than for watermarking applications—it is not uncommon for steganographic algorithms to embed thousands of bits. Fortunately, this requirement is made somewhat easier by the fact that Alice is free to select which cover Work to use. Thus, Works in which it is difficult to conceal a message can be avoided. However, digital watermarking has no such control over the selection of Works.

When designing a steganographic scheme, we need to consider issues such as the properties of the communication channel, the source of cover Works, and the embedding/extraction function. In Sections 12.1 and 12.2, we discuss these issues and define basic terminology that is commonly used to describe and evaluate steganographic schemes.

The central concept in steganography is statistical undetectability. Without a precise definition, the field of steganography would lack the criterion to evaluate how secure steganographic schemes really are. In Section 12.3, we provide an information-theoretic framework for steganography, which provides a rigorous definition for *undetectability*. The information-theoretic concepts are illustrated using one of the simplest algorithms for digital steganography, which hides a message in the least significant bit (LSB) of a Work.[1]

The theoretical foundations described in Section 12.3 are reflected in Section 12.4, where we discuss practical steganographic schemes that aspire to satisfy the requirements for statistical undetectability. We describe a general framework for constructing steganographic schemes based on a statistical model of the cover Works. We also mention other approaches that attempt to build statistically undetectable schemes using heuristic principles, such as masking the embedding modification as a naturally ocurring process.

In Section 12.5, we cast the embedding process in terms of coding theory. This formulation enables us to minimize the number of embedding changes (matrix embedding) and construct steganographic schemes with arbitrary selection rules (wet paper codes) (i.e., the detector does not need to know where the embedder altered the cover Work).

The complementary task to steganography is steganalysis—discovering the *presence* of steganographic channels. Steganography is considered broken if even the presence of secretly embedded data is detected (i.e., it is not

[1] As with almost all of this book, we focus on digital images, but the concepts can be easily generalized to other forms of digital content.

necessary to decode the message). The task of recovering some attributes of the message or the stego method used is called *forensic steganalysis*. Steganalysis can be divided into two categories—targeted and blind—depending on whether the attack uses the knowledge of the steganographic algorithm. The subject of steganalysis is treated in Chapter 13.

12.1 STEGANOGRAPHIC COMMUNICATION

The first informal definition of a steganographic scheme was formulated by Simmons [375] as the Prisoners' Problem. Two prisoners, Alice and Bob, are under the surveillance of a warden, Eve. The warden will permit Alice and Bob to communicate, but all communications must go through the warden. If the warden thinks that Alice's message to Bob is innocuous, she may simply forward it to Bob. Alternatively, she may intentionally distort the content (e.g., apply lossy compression) in the hope that such a distortion will remove any secret message that just might be present. If the warden thinks Alice's message to Bob hides a covert communication, then she may block the communication entirely.

This framework, which models the applications discussed in Chapter 2, is depicted in Figure 12.1. A number of different assumptions can be made regarding the channel, the source of cover Works, and the embedding and extraction functions. In the next section, we elaborate on the properties of the channel, while the subsequent section focuses on the basic properties of the embedding and extraction functions.

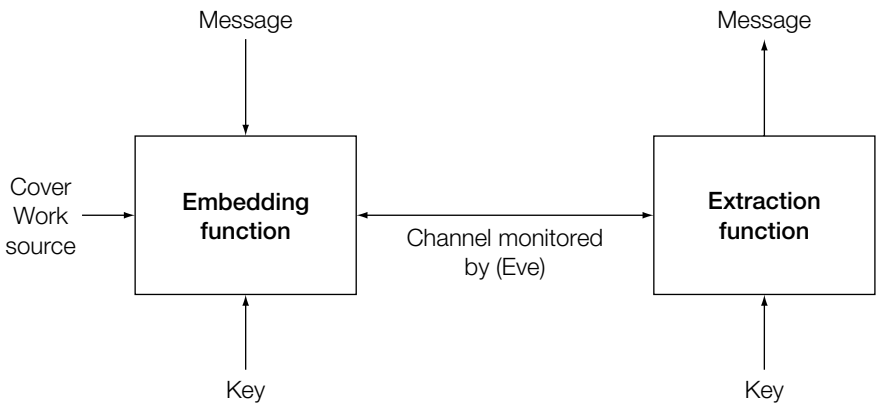


FIGURE 12.1

Steganographic embedding scheme.

12.1.1 The Channel

In steganography, the physical channel used for communication is generally assumed noise free, as this can be ensured using error correction and standard Internet protocols. Instead, the channel's properties are defined by the warden. The warden is considered a part of the channel because she may, or may not, interfere with the communication. As such, there are three types of warden: passive, active, and malicious.

The warden is called *passive* if she is restricted from modifying the content sent by Alice prior to receipt by Bob (i.e., the warden can only prevent or permit delivery of Alice's message). In this scenario, the warden tests each communication from Alice for the presence of a covert message. If the warden's test is negative, the communication is relayed to Bob. Otherwise it is blocked. This is the most commonly assumed scenario and why most steganographic algorithms are not designed to be robust.

The warden is called *active* if she intentionally modifies the content sent by Alice prior to receipt by Bob. In this scenario, the warden may not be entirely confident of her steganalysis program. Thus, even though her tests are negative, the warden may alter the content, hoping that the modification will destroy any steganographic message that *might* be present. If the steganographic algorithm assumes a passive warden, then there is a good chance that alterations to the content will severely degrade or remove the hidden message. The types of modification an active warden might apply include lossy recompression of images and audio clips, low-pass filtering, and other procedures that slightly degrade the content.

The warden is called *malicious* if her actions are based on the specifics of the steganographic scheme and are aimed at catching the prisoners communicating secretly. This may include the warden trying to impersonate Alice or Bob or otherwise tricking them. A malicious warden is usually considered in public-key steganography [22, 430]. In this scenario, the *stego* key is known and anyone can extract the secret message. However, the message is encrypted using a public-key cryptosystem. Only those who possess Bob's private key can decipher Alice's message. Even though the stego key is known, it is difficult to distinguish between an encrypted message and a random bit sequence extracted from a cover Work. Nevertheless, since the Warden also knows the stego key, she has more options to attack the stego system [26, 100].

In this chapter we will not consider cases where the warden is active or malicious, since the vast majority of research in steganography is concerned with the passive warden scenario. We focus on steganography in digital media, because this field is the most developed today, ignoring other less-developed topics, such as linguistic steganography [441] or data hiding in network protocols [277]. The subject of subliminal channels is covered in the cryptographic literature [374].

In the next section we describe various choices Alice and Bob have when designing a steganographic scheme. We also define basic terminology that is commonly used to describe and evaluate steganographic schemes.

12.1.2 The Building Blocks

The main building blocks of any steganographic algorithm are:

1. The choice of the cover Work.
2. The embedding and extracting algorithms, which might include
   a. Symbol assignment function.
   b. The embedding modification.
   c. The selection rule.
3. Stego key management.

We now discuss each of these design elements in more detail.

Unlike a watermark, a steganographic message says nothing about the cover Work in which it is hidden. Consequently, the steganographer is free to choose a particular cover Work from his or her source of covers. The main restriction is the source of cover Works, which is determined by the resources available to Alice and Bob, by the warden herself, and the context in which the communication takes place. For example, an oppressive regime (Section 2.2.1) can specify allowable forms of messages and Alice must comply with them to avoid being caught. Or, if Alice and Bob communicate by posting images to a discussion newsgroup, they must choose the covers among those that are typically posted. But even with these restrictions, there are still numerous cover Works in which to hide the covert message. Alice is therefore at liberty to choose the cover Work which, after embedding, has the least likelihood of being detected. For example, it is intuitively clear that noisy or highly textured images will better mask any embedding changes than high-quality images with little content (e.g., blue sky images). Alternatively, Alice can think ahead and attempt to guess what tests the warden is going to use and embed the same message into many different covers, run known steganalysis attacks on each stego Work, and then simply send the cover that passes the tests [233].[2]

[2] In fact, it is possible to go a step further and choose the cover Work such that it is correlated with the hidden message. For example, if the hidden message is an image, choose an image that is similar. The advantage of this is that the minimum number of bits needed to encode the hidden message is now the conditional entropy of the hidden image given the cover. This may be very much smaller than the entropy of the hidden image itself. As we will see, the fewer bits that we need to hide, the less likely the warden will detect the stego Work. Furthermore, the information transferred by Alice to Bob can now be much greater than the number of bits embedded. This is because the cover Work is now providing Bob with additional information. The interested reader is directed to [92, 460] for further details.

Given the source of cover Works (e.g., a class of images or songs), Alice and Bob need to construct the embedding and extraction functions. In watermarking, it is not uncommon to specify the detector design but not the encoder design, since there may be a variety of ways to perform the encoding, each with different tradeoffs (e.g., computational and economic costs). This design principle is absent from steganography, where both the embedding and extracting functions are jointly defined.

Fundamentally, an embedding function can be based on three different principles, namely:

1. The cover Works are preexisting and the embedder does *not* modify the cover Works. This is referred to as steganography by cover lookup.
2. The cover Works are generated based on the hidden message and the embedder does *not* modify the cover Works. This is referred to as cover synthesis.
3. The cover Works are preexisting and the embedder modifies the cover Works. This is referred to as steganography by cover modification.

How can we send a message without modifying the cover Work? Consider the case where Alice only needs to send, say, 10 bits to Bob. Then, if Alice has a collection of approximately 1,000 songs, she can determine which song, concatentated with their shared key, hashes to the desired 10-bit message. Hence, steganography by cover lookup. Since the song has not been modified in any way, it is almost impossible for the warden to determine that a steganographic message is present. Of course, as the required payload increases, this solution quickly becomes impractical. To send 20 bits requires a million songs, and to send 30 bits requires a billion songs. Although Alice might solve this problem by sending 10k bits at a time, a 10k-bit message would require the transmission of 1,000 songs, which might raise the suspicion of the warden.

In steganography by cover synthesis, Alice *creates* the stego Work without recourse to a cover Work. An interesting real-world example of such a system has been described at the end of *Between Silk and Cyanide*, by Leo Marks [284]. The code, called "Windswept," which was used by British spies in World War II, works in the following manner. The code consisted of a big book of conversations with alternate wordings for each line, and even alternate courses that the conversation could take. Each line was associated, arbitrarily, with a number. By selecting different phrases from the book, British spies could thus encode sequences of numbers in perfectly innocuous conversations. The code had fairly high capacity because the communicating parties did not care at all about the content of the conversations and thus could change it in any way necessary to suit the secrets.

Another method to synthesize the stego Work uses so-called mimic functions [440]. The program SpamMimic (*http://www.spammimic.com*) will encode a short message into a document that resembles a typical spam. Basically, the sentences produced by the program are a function of the secret message. Because spammers use unusual spellings, punctuation, and sentence structures, it is rather hard to distinguish spam containing a hidden message from regular spam.

A further example of steganography by synthesis is called data masking [341]. Here the goal is to shape a secret message into a signal whose statistical properties resemble those of, say, music. If the signal indeed shares essential statistical properties with typical music files, then any automatic steganalysis engine that checks for statistical anomalies will not detect anything suspicious. Of course, this method of steganography will not be successful if the warden listens to the music that Alice and Bob exchange.

Steganography by cover modification describes methods where Alice alters an existing cover Work to create a stego Work that conveys the desired message. This approach is both the most common and the most advanced. In this book, we will only focus on this class of steganographic algorithms.

The type of changes introduced by the embedder, together with the location of these changes within the cover Work, have a major influence on how inconspicuous the embedded message will be. Intuitively, changes of large magnitude will be more obvious than changes of smaller magnitude. Consequently, most steganographic schemes try to modify the cover Work as little as possible.

The location of the changes is controlled by the *selection rule*. There are three types of selection rules: sequential, (pseudo) random, and adaptive.

A sequential selection rule embeds the message bits in individual elements of the cover Work in a sequential manner, for example, starting in the upper left corner of an image and proceeding in a row-wise manner to the lower right corner. Although the sequential selection rule is the easiest one to implement, it provides poor security, since steganalysis algorithms can inspect the statistical properties of pixels in the same order, looking for a sudden change in behavior (see Section 13.2.1).

A pseudo-random selection rule embeds the message bits in a pseudo-randomly selected subset of the cover Work. The sender might first use a secret stego key, K_s, to initialize a pseudo-random number generator (PRNG) that in turn generates a pseudo-random walk through the cover Work. The message bits are then embedded into the elements constituting this walk. Pseudo-random selection rules typically offer better security than sequential rules.

An adaptive selection rule embeds the message bits at locations that are determined based on the content of the cover Work. The motivation for this is that statistical detectability is likely to depend on the content of the cover

Work as well. For example, it will be more difficult to detect embedding changes in noisy images or in highly textured areas of the image compared with smooth, uniform areas. Thus, one may desire to adjust the selection rule to the specific content of the cover Work. For example, consider LSB embedding once more. The selection rule could depend on the variance of pixels within a small local neighborhood. Only pixels whose local neighborhood variance exceeds a certain threshold would be candidates to be modified. The influence of the selection rule on the security of steganographic schemes are discussed in more detail in Section 12.5 where we introduce wet paper codes.

The process of embedding is controlled by a secret key shared between Alice and Bob. The key can be used for several different purposes. As previously mentioned, the key may seed a pseudo-random number generator to generate a random walk through the cover Work. It can also be used to generate other pseudo-random entities needed for embedding. For example, in stochastic modulation steganography (see Section 12.4.3), the message is embedded by adding a noise signal with specific statistical properties to the cover Work. The generation of this signal may depend on the stego key.

Alice and Bob do not have to use, and in fact should not use, the same stego key for every cover. Imagine that the covers are all images of the same dimensions. Then, a fixed stego key would produce the same pseudo-random embedding walk in every cover. The warden could use the fact that the embedding changes between different covers are correlated to mount an attack. Thus, Alice and Bob may wish to adopt a more sophisticated key management and periodically change the key according to some pre-agreed protocol. For example, the message may be communicated using a session key that is different for each cover and communicated in the cover itself.

It is important to choose strong stego keys, otherwise the warden could attack the steganographic scheme simply by trying to read messages from the stego Work using all possible stego keys. The correct key would be revealed when a meaningful message is obtained. Although this attack would not work if the message was encrypted before embedding, there exist more advanced versions of this attack [151]. In general, it is always a good practice to encrypt the message before embedding. The crypto key could be derived from the stego key or could be chosen independently. The second choice provides better security in cases where the stego key is compromised.

The primary goal of steganography is to design embedding functions that are statistically undetectable and capable of communicating practical (i.e., large) payloads. In order to do so, we need to first define in mathematical terms what we mean by statistical undetectability. This is the subject of Section 12.3. However, before proceeding to do so, we briefly introduce some notation and terminology.

12.2 NOTATION AND TERMINOLOGY

We now mathematically define a steganographic scheme. Let K_s denote a stego key drawn from a set, $\mathcal{K}$, of all secret stego keys, $\mathcal{M}$ the set of all embeddable messages, and $\mathcal{C}$ the set of all cover Works. A steganographic scheme is formed by two mappings, the embedding mapping, *Emb*, and the extraction mapping, *Ext*:

$$Emb: \mathcal{C} \times \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$$

$$Ext: \mathcal{C} \rightarrow \mathcal{M}, \tag{12.1}$$

such that $Ext(Emb(\mathbf{c}, K_s, \mathbf{m})) = \mathbf{m}$ for all $\mathbf{c} \in \mathcal{C}$, $K_s \in \mathcal{K}$, and $\mathbf{m} \in \mathcal{M}$. The Work $\mathbf{s} = Emb(\mathbf{c}, K_s, \mathbf{m})$ is called the stego Work.

The embedding algorithm *Emb* takes the cover Work, the secret key, and the message as its input and produces the modified stego Work. For a given embedding function, the cardinality, $|\mathcal{M}|$, of the set, $\mathcal{M}$, is the number of different messages that can be embedded in a specific Work. The logarithm, $\log_2 |\mathcal{M}|$, is called the *embedding capacity* and its units are bits. For example, for LSB embedding in a grayscale image, the embedding capacity is equal to the number of pixels. If the length, m, of the embedded message is smaller than the embedding capacity, we say that the message has *relative length*, $m/|\mathcal{M}|$.

The impact of embedding (or the embedding distortion) is measured as $D(\mathbf{c}, \mathbf{s})$, where D is a distance defined on $\mathcal{C}$. A commonly used measure of distance is mean square error, as discussed in Chapter 8. Intuitively, steganographic schemes that introduce smaller embedding distortion are less likely to introduce statistically detectable artifacts than schemes with larger distortion. The average number of embedded bits per unit distortion is called the *embedding efficiency*. We study this concept in more detail in Section 12.5.

12.3 INFORMATION-THEORETIC FOUNDATIONS OF STEGANOGRAPHY

In 1949, Shannon [365] wrote a seminar paper entitled "Communications Theory of Secrecy Systems," within which he provided a precise definition of cryptographic security. Interestingly, Shannon identified three forms of secret communications: (1) "*concealment systems, including such methods as invisible ink, concealing a message in an innocent text, or in a fake covering cryptogram, or other methods in which the existence of the message is concealed from the enemy*," (2) privacy sytems, and (3) cryptographic systems. However, with regard to concealment systems (i.e., steganography), Shannon declared that such "*systems are primarily a psychological problem*" and did not consider them further.

Is it possible to give steganographic security the same precise meaning as Shannon gave to information security? In this section, we attempt to give the reader at least a partial answer to this question and to explain why the answer continues to be debated by researchers.

To the best of our knowledge, it was not until 1996 that the application of information theory to steganography was revisited. Anderson and Petitcolas [20, 22] observed that if perfect compression existed, then it would be possible to transform any arbitrary bitstream into a legitimate cover Work in the following manner. Consider a perfect compressor of music. If such a compressor existed, then every perceptually different music file would be represented by a unique code word. And these code words would appear to be random bit sequences. More important, the associated decompressor would convert any random bit sequence to a perceptually acceptable musical piece. Thus, if Alice passes her encrypted hidden message through this perfect *decompressor*, the result will be music. If the warden permits Alice to send any form of music to Bob, then no modifications to an existing piece of music are necessary. This is a form of embedding based on cover synthesis. Finally, Bob can compress the received music file through the perfect compressor in order to obtain the original encrypted message.

Of course, the chances of a random bit string mapping to a known musical piece is unlikely, and the warden may become suspicious if she does not recognize any of the musical pieces that Alice is sending to Bob. Notice that the fact that Alice and Bob never modify their Works does not necessarily imply that the stego method is undetectable. To explain why this does not have to be so, imagine that Alice sends only one out of a small number of possible messages. For example, either "attack" or "do not attack." Then, over time she would be resending the same pair of music files, which would be very suspicious. She can attempt to alleviate this problem by appending a string of dummy random bits to her messages so that the messages are mapped onto a larger set of musical files. However, even if Alice appends 128 random bits she still cannot be sure if the statistical distribution of her stego musical files is free of obvious artifacts. In other words, Alice needs to be concerned with the statistical distribution of her stego Works and how closely it matches some "natural" distribution. The question then becomes, how much information can Alice hide in a Work chosen from a distribution of cover Works, while ensuring that the probability of detection is negligibly small? This observation forms the foundation of the most widely used definition of steganographic security due to Cachin.

12.3.1 Cachin's Definition of Steganographic Security

Cachin's definition of steganographic security [60] assumes that the warden will permit Alice to send any cover Work, $\mathbf{c}$, to Bob, provided it is drawn from a probability distribution, P_C. And $P_C(\mathbf{c})$ is the probability of drawing a cover Work, $\mathbf{c}$, from this distribution.

Before proceeding, it is worth considering what this assumption means. Qualitatively, the probability distribution, P_C, represents Eve's knowledge of what types of transmissions between Alice and Bob are legitimate. For example, it might be that Eve expects Alice and Bob to only transmit family photos. Thus, provided a photograph, $\mathbf{c}$, contains an image of one or more family members, then the image has a finite probability, $P_C(\mathbf{c})$, of being generated from this distribution. Conversely, a photograph of Queen Elizabeth would have a very low or zero probability of deriving from the assumed distribution, P_C, and therefore a very high likelihood, in Eve's mind, of being suspicious.

In practice, it may be very difficult to quantify this distribution of "natural" images. However, it may be feasible for some specific steganographic methods to describe the opposite—the distribution of Works carrying a secret message. In other words, even though we may not be able to tell if an image is "natural," we may be very certain that it is not, because it contains anomalies that are characteristic of a certain steganographic method. We illustrate this point with an example of the most commonly used steganographic method, called LSB embedding.

In LSB embedding, the message bits are encoded as the LSBs of pixel values.[3] For example, to embed a 0 at a pixel with grayscale value 51, we first write the value as a binary number $(51)_2 = 00110011$, the last bit being the LSB and the first bit the most significant bit (MSB), and then replace the LSB with the message bit. In this case, the binary representation is modified to $00110010 = (50)_2$. Obviously, if we were to embed a 1, no change would be necessary. As a result of our embedding, the value 51 was changed to 50. It is easy to see that the values 50 and 51 form a pair, such that during embedding these two values can be changed into each other but never to any other values. Thus, the set of all possible pixel values can be divided into disjoint pairs of values (PoVs), $(2i, 2i + 1)$, $i = 0, \ldots, 127$, that may be changed into each other during embedding.

The asymmetry of the embedding function introduces a characteristic artifact into the first-order statistics (the histogram) of the stego Work that practically never occurs in real imagery. In Figure 12.2, we show a portion of the histogram of a grayscale cover image before and after the cover has been fully embedded with LSB embedding (1 bit per each pixel). We can clearly see that the histogram bins of each PoV, $(2i, 2i + 1)$, have been equalized by embedding. This is, in fact, intuitively clear, because in an image fully embedded with LSB embedding, the LSBs have been completely randomized and thus any pixel from the PoV is equally likely to end up with value $2i$ as

[3] We note that LSB embedding can be applied to any collection of numerical samples, such as pixel values, audio samples, palette indices (for GIF images), or quantized DCT coefficients for JPEG images.

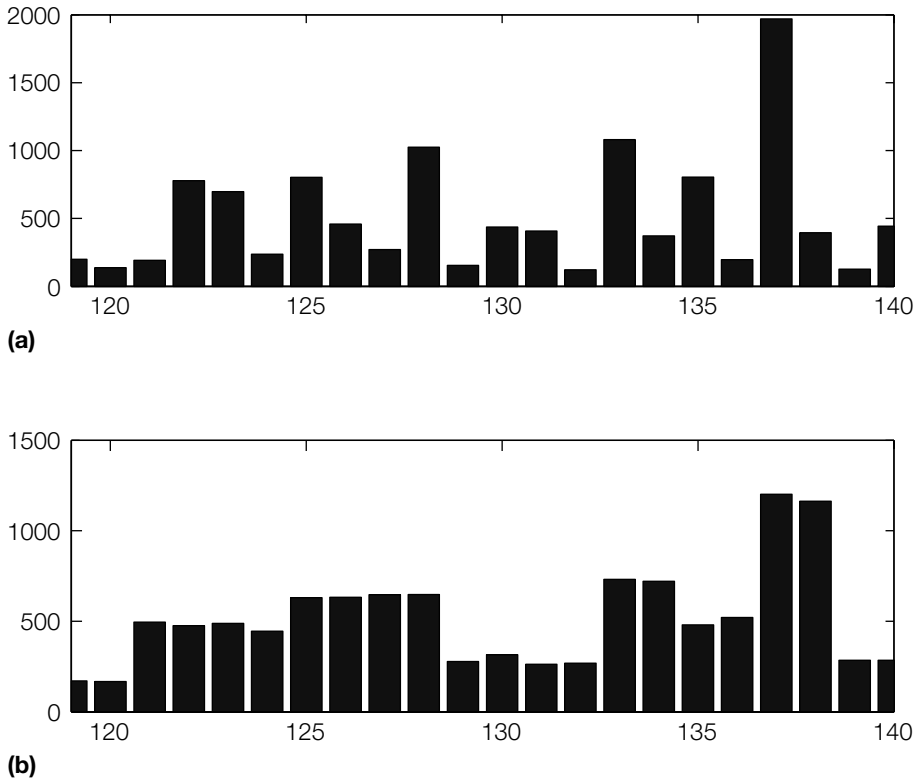


FIGURE 12.2

(a) A portion of a cover image histogram. (b) The same portion of the histogram of an image fully embedded with LSB embedding. Note the characteristic steps in the histogram.

with value $2i + 1$. This observation is the basis of the histogram attack described in Section 13.2.1.

Let us now try to mathematically formalize the concept of steganographic security. We will use the definition of a steganographic scheme given in Equation 12.1.

A steganographic scheme is formed by two mappings: the embedding mapping, *Emb*, and the extraction mapping, *Ext*. The embedding function assigns a stego Work, $\mathbf{s}$, to each combination of the cover Work, $\mathbf{c} \in \mathcal{C}$, stego key, $K_s \in \mathcal{K}$, and message, $\mathbf{m} \in \mathcal{M}$. Assuming the cover Work is drawn from the space of all possible cover Works with probability distribution, $P_{\mathcal{C}}$, and the stego key as well as the message are uniformly distributed over their corresponding spaces, $\mathcal{K}$, $\mathcal{M}$, the distribution of stego Works will be denoted as $P_{\mathcal{S}}$.

Cachin analyzed the warden's performance using the theory of hypothesis testing. Thus, given a Work, $\mathbf{c}$, Eve must decide between two hypotheses: H_0 represents the hypothesis that the Work *does not* contain a hidden message and H_1 represents the hypothesis that the Work *does* contain a hidden message. If hypothesis H_0 is true then the observation, $\mathbf{c}$, is drawn from the distribution, $P_{\mathcal{C}}$. Conversely, if hypothesis H_1 is true then the observation, $\mathbf{c}$, is drawn from the distribution, $P_{\mathcal{S}}$.

Comparison of the two distributions can be made based on their relative entropy, also known as the Kullback-Leibler distance, between two probability distributions, defined as

$$D(P_{\mathcal{C}} \| P_{\mathcal{S}}) = \sum_{\mathbf{c} \in \mathcal{C}} P_{\mathcal{C}}(\mathbf{c}) \log \frac{P_{\mathcal{C}}(\mathbf{c})}{P_{\mathcal{S}}(\mathbf{c})}. \tag{12.2}$$

The relative entropy is always non-negative and is equal to 0 if and only if $P_{\mathcal{C}} = P_{\mathcal{S}}$. Also note that for probability distributions on a set containing only two elements, $P = (p_1, 1 - p_1)$ and $Q = (q_1, 1 - q_1)$, Equation 12.2 becomes the binary relative entropy defined as

$$D(P \| Q) = p_1 \log \frac{p_1}{q_1} + (1 - p_1) \log \frac{1 - p_1}{1 - q_1}. \tag{12.3}$$

Although relative entropy is not a distance in the strict mathematical sense, because it is nonsymmetrical and does not satisfy the triangle inequality, it is useful to think of it as a distance. The relative entropy can be thought of as the difference between Huffman coding a source drawn from a pdf, P, using a table determined by the true distribution, P, and an alternative table determined by another distribution, Q (the latter leads to a suboptimal choice of code word lengths).

When $D(P_{\mathcal{C}} \| P_{\mathcal{S}}) = 0$, that is, the distribution of the stego Works, $P_{\mathcal{S}}$, created by Alice is identical to the cover distribution, $P_{\mathcal{C}}$, assumed by the warden, Alice's stegosystem is perfectly secure, as defined by Cachin. This is because it is impossible for the warden to distinguish between cover Works and stego Works. If $D(P_{\mathcal{C}} \| P_{\mathcal{S}}) \leq \epsilon$, then Cachin defined the system as ϵ-secure.

To understand what is meant by ϵ-secure, consider that Eve, the warden, can make two forms of error. The first type of error (type I) is a false positive that occurs when Eve decides that a hidden message is present when in fact it is absent. The second type of error (type II) is a false negative that occurs when Eve decides that a hidden message is absent, when in fact it is present. Let α and β denote the probabilities of type I and type II errors, respectively.

The response of Eve's detector is binary—it answers either 0 for cover or 1 for stego. Thus, assuming the detector receives only cover Works with probability P_C, Eve will decide 0 or 1 with probabilities $p_1 = 1 - \alpha$ and $p_2 = \alpha$, respectively. On stego Works distributed according to $P_{\mathcal{S}}$, Eve's detector assigns 0 and 1 with probabilities $q_1 = \beta$ and $q_2 = 1 - \beta$, respectively.

The relative entropy given by Equation 12.3 between both distributions of Eve's detector is

$$d(\alpha, \beta) = (1 - \alpha) \log \frac{1 - \alpha}{\beta} + \alpha \log \frac{\alpha}{1 - \beta}. \tag{12.4}$$

A standard result in information theory states that the relative entropy of processed data can never increase (see, for example, Chapter 2 of Cover and Thomas [89]). Because Eve's detector is a type of processing, we must have

$$d(\alpha, \beta) \leq D(P_C \parallel P_S). \tag{12.5}$$

This inequality can be used to determine a lower bound on the probability of false negatives (type II error), β, given a desired upper bound on the probability of a false positive (type I error), α. In particular, if the probability of a type I error is $\alpha = 0$ (i.e., Eve is not permitted to accuse Alice of transmitting a covert message when in fact she has not), then setting $\alpha = 0$ in Equation 12.5 gives $\log \frac{1}{\beta} \leq D(P_C \parallel P_S) \leq \epsilon$. In other words, the probability of a type II error, β (i.e., of missing a covert communication) is

$$\beta \geq 2^{-\epsilon}. \tag{12.6}$$

Thus, the smaller ϵ is, or the closer the two distributions, P_C and P_S, are, the greater the likelihood that a covert communication will *not* be detected.

Steganographic systems that are secure in Cachin's sense indeed exist. Consider the following simple one-time pad steganographic system with $\mathcal{C} = \{0, 1\}^n$ and P_C the uniform distribution on $\mathcal{C}$. Given a secret message, $\mathbf{m} \in \{0, 1\}^n$, the sender selects $\mathbf{K}_s \in \mathcal{K} = \{0, 1\}^n$ at random and computes the stego object as the XOR (exclusive-or) $\mathbf{s} = \mathbf{K}_s \oplus \mathbf{m}$. The message is extracted by the recipient as $\mathbf{m} = \mathbf{K}_s \oplus \mathbf{s}$. This system is, however, not very useful because no warden will allow the exchange of random bit strings. For construction of other provably secure steganographic schemes, the reader is referred to [201, 234].

The information-theoretic definition of steganographic security does not consider issues of practical realizability and computational complexity. One could base the concept of steganographic security on the inability of the warden to carry out the necessary calculations to gather sufficient evidence that Alice and Bob communicate secretly. The definition proposed by Katzenbeisser and Petitcolas [222] takes into account computational complexity and is based on a probabilistic game played between a third party (a judge) and the warden. Another possibility is to define steganographic security with respect to a specific detector [70]. The capacity of steganographic channels with noise was studied in Harmsen and Pearlman [175]. Assuming there exist bounds on the maximal admissible distortion introduced by both the warden, Alice, and Bob, the problem of how much information can be reliably and securely communicated can be formulated in terms of game theory [131, 300, 303, 304, 436].

It is likely that the concept of steganographic security will follow the same path as security in cryptography. The only unconditionally secure cryptographic system is the one-time pad, which is, however, rarely used because of its impracticality. Cryptographic schemes, such as DES (Data Encryption Standard), cannot be proved secure but are nevertheless widely used because of their practical advantages. Their security lies in the fact that nobody has so far been able to produce an attack substantially faster than brute-force search for the key. Similarly, we may consider a steganographic system "practically secure" if no existing attacks can be modified to mount a successful attack on the steganographic scheme. In practice, security of specific embedding algorithms is usually evaluated using existing targeted staganalyzers and blind steganalyzers (see Section 13.1.1).

12.4 PRACTICAL STEGANOGRAPHIC METHODS

All practical steganographic methods try, in one way or another, to comply with Cachin's definition of steganographic security. This can be achieved either by postulating some heuristic principles and designing an embedding scheme that follows these heuristics or by finding a simplified model for the space of Works, $\mathcal{C}$, and making sure that the steganographic embedding preserves this model. We start with the second approach.

12.4.1 Statistics Preserving Steganography

Because the dimensionality of the space of Works, $\mathcal{C}$, is approximately determined by the number of pixels, it is very difficult to obtain even a rough approximation to the distribution, $P_{\mathcal{C}}$. We can, however, attempt to represent each Work in a much lower dimensional space. For example, we can assume that each Work is completely described by its histogram. This would, indeed, be the case if the colors in an image were realizations of independent indentically distributed (i.i.d.) random variables, which, of course, they are not. Under this assumption, a steganographic system that preserves the histogram of the cover Work would be secure.

In the next section, we describe one of the first steganographic schemes that aims to exactly preserve the histogram of 8 × 8 DCT coefficients in a cover Work. Because DCT coefficients in an individual 8 × 8 block are largely decorrelated and because interblock dependencies among DCT coefficients are less strong than dependencies among neighboring pixels in the spatial domain, the DCT coefficients can be approximately modeled as an i.i.d. signal. Of course, if the warden has a more sophisticated model, then preserving the first-order statistics of DCT coefficients may not be sufficient to avoid detection, as shown in Section 13.2.3.

Preserving DCT Statistics: OutGuess

The OutGuess algorithm [338] is a two-pass procedure. In the first pass, OutGuess embeds message bits along a pseudo-random walk of the LSBs of DCT coefficients, skipping coefficients whose magnitudes are zero or unity. In the second pass, corrections are made to the magnitudes of the coefficients in order to ensure that the histogram of the DCTs of the stego image match those of the cover image.

Prior to embedding, OutGuess calculates the maximum length of a randomly spread message that can be embedded in the image during the first pass, while ensuring that one will be able to make corrections to adjust the histogram to its original values during the second pass. Let n_{01} be the number of all DCT coefficients whose magnitudes are not equal to 0 or 1. Because the majority of DCT coefficients are zeros, modifying them would introduce a large visible distortion. This is why OutGuess does not embed in zeros. Since 0 is paired with 1 in their LSB pair, coefficients equal to 1 are also skipped. After some thought, it is clear that the maximal correctable message length is determined by the frequencies of the most unbalanced LSB pair. Let us denote the histogram of DCT coefficients as $\mathbf{T}_c$. Due to the shape of the DCT histogram, the most unbalanced LSB pair is the pair $(-2, -1)$ (remember that the pair $(0, 1)$ is not used). Because $\mathbf{T}_c[-2] < \mathbf{T}_c[-1]$, after embedding the maximum correctable message we will need to move all remaining coefficients with value -2 to -1 in the second phase.

To obtain the maximum *correctable* relative embedding capacity, q, we start with the expression for the expected value of the histogram of DCT coefficients after embedding. Because the embedding mechanism is LSB embedding, we can use Equation 13.1, in Chapter 13 (p. 478), with qn_{01} as the length of the embedded message. The number of unused DCT coefficients with value -2 after embedding is $(1 - q)\mathbf{T}_c[-2]$, and this must be larger than the decrease in the number of coefficients with value -1, which is $\frac{q}{2}\mathbf{T}_c[-1] - \frac{q}{2}\mathbf{T}_c[-2]$, where the first term is the number of DCT coefficients with value -1 that are changed *from* the value -1 and the second term is the number of DCT coefficients that are altered *to* the value -1. Thus, we obtain the following inequality and eventually an upper bound on the correctable message length q:

$$(1 - q)\mathbf{T}_c[-2] \geq \frac{q}{2}\mathbf{T}_c[-1] - \frac{q}{2}\mathbf{T}_c[-2]$$

$$q \leq \frac{2\mathbf{T}_c[-2]}{\mathbf{T}_c[-1] + \mathbf{T}_c[-2]}.$$

This condition guarantees that on average there will be enough unused coefficients with magnitude -2 that can be flipped back to -1 to make sure that the frequencies of the LSB pair $(-2, -1)$ are preserved after embedding. Since this pair is the most unbalanced one, virtually all other pairs can be preserved using the same correction step, as well. We note that some very sparsely populated histogram bins in the tails of the DCT histogram may not be adjusted correctly

during the second phase, but since their numbers are statistically insignificant, the impact on statistical detectability is negligible.

Other approaches to histogram-preserving steganography for JPEG imagery can be found in [123, 193, 311, 381, 413].

Because neighboring pixels in the spatial domain are much more correlated than DCT coefficients in a JPEG file, steganographic schemes that embed messages in the spatial domain must preserve more than the histogram of pixels and thus require more complex models. Because first-order statistics cannot capture the correlations among neighboring pixels, the joint statistics of neighboring pixel *pairs* [136] or Markov chains [373, 394] may be used instead.

12.4.2 Model-Based Steganography

Model-based steganography is a general framework for constructing steganographic systems that preserve a chosen *model* of the cover Works [357] rather than its statistics. The cover Work, c, is modeled as a random variable that can be split into two components, $(\mathsf{c}_{inv}, \mathsf{c}_{emb})$, where c_{inv} is invariant to embedding and c_{emb} may be modified during embedding. For example when using LSB embedding, c_{inv} and c_{emb} correspond to the 7 most significant bits and the least significant bit of DCT coefficients, respectively.

The designer of the stego system first chooses a model for cover Works, which will be represented by conditional probabilities $P(\mathsf{c}_{emb}|\mathsf{c}_{inv})$. It is important that these probabilities can be calculated from the invariant portion only so that the same information is available to both the sender and the recipient. For each value $\mathbf{c}_{inv}$, let $\mathcal{L}(\mathbf{c}_{inv})$ be the set of pixels (or DCT coefficients) with their invariant part equal to $\mathbf{c}_{inv}$. The uniformly distributed message bits are run through an entropy decoder (e.g., an arithmetic decompressor with 0s and 1s occurring with probabilities $P(0|\mathsf{c}_{inv})$ and $P(1|\mathsf{c}_{inv})$) to change the message's uniform distribution to the required distribution expressed by the conditional probabilities. Thus, by replacing $\mathbf{c}_{emb}$ for pixels/DCT coefficients in $\mathcal{L}(\mathbf{c}_{inv})$ with the transformed message bits, we obtain a stego Work with similar statistical properties. The embedding and extraction procedures are illustrated in Figures 12.3 and 12.4.

The fraction of bits we can embed in each element of $\mathcal{L}(\mathbf{c}_{inv})$ is the entropy of $P(\mathsf{c}_{emb}|\mathsf{c}_{inv} = \mathbf{c}_{inv})$ or

$$H(P(\mathsf{c}_{emb}|\mathsf{c}_{inv} = \mathbf{c}_{inv})) = -\sum_{\mathbf{c}_{emb}} P_{\mathsf{c}_{emb}|\mathsf{c}_{inv}}(\mathbf{c}_{emb}|\mathbf{c}_{inv}) \log_2 P_{\mathsf{c}_{emb}|\mathsf{c}_{inv}}(\mathbf{c}_{emb}|\mathbf{c}_{inv}). \tag{12.7}$$

Thus, the total embedding capacity is

$$\sum_{\mathbf{c}_{inv}} |\mathcal{L}(\mathbf{c}_{inv})|\, H(P(\mathsf{c}_{emb}|\mathsf{c}_{inv} = \mathbf{c}_{inv})). \tag{12.8}$$

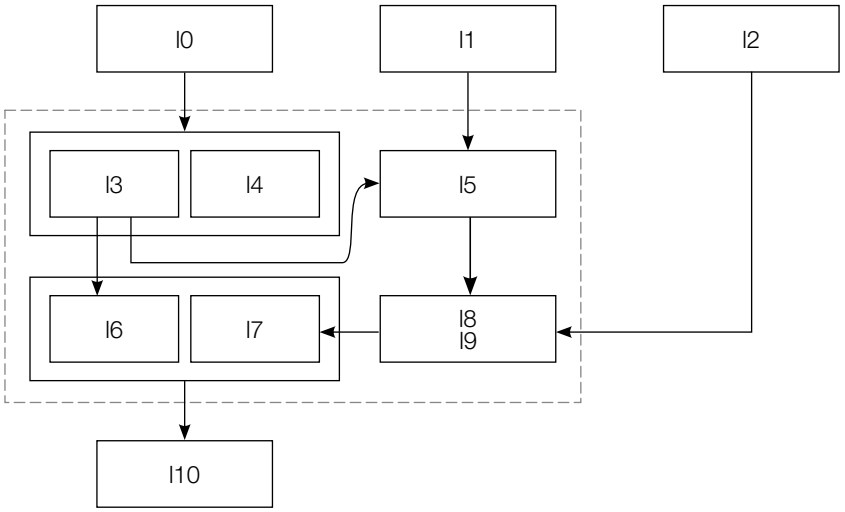


FIGURE 12.3

Embedding scheme for model-based steganography.

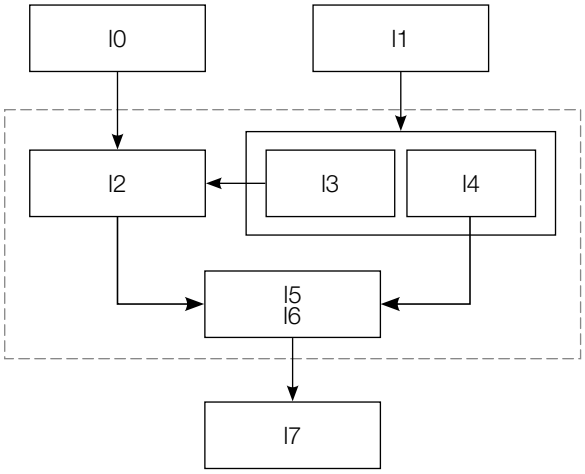


FIGURE 12.4

Extracting algorithm for model-based steganography.

To illustrate the model-based approach, let us consider a specific realization of this approach for JPEG images [357]. As with OutGuess, DCT coefficients with magnitudes of 0 or 1 are not used for embedding. The cover JPEG file is decomposed into 64 subsets corresponding to the 64 DCT frequencies. Let

us examine a specific DCT frequency, for example (2, 2), which corresponds to the fifth DCT coefficient in the zig-zag scan. Let $\mathbf{T}_c[i]$ be the histogram of all (2, 2) DCT coefficients in the cover image. The embedding mechanism will again be LSB embedding. Because LSB embedding preserves the sums $\mathbf{S}[2i] = \mathbf{T}_c[2i] + \mathbf{T}_c[2i+1]$ for all i, we can use this invariant and fit a parametric model through the points $(2i, \mathbf{S}[2i])$. For example, we can model the DCT coefficients using the generalized Cauchy model with pdf

$$\frac{p-1}{2s}|x/s + 1|^{-p},$$

and use a maximum likelihood estimator to determine the two model parameters, $p > 1$ and $s > 0$. By sampling this model at integer values, we can read out the expected values of the histogram at $\mathbf{T}_c[2i]$ and $\mathbf{T}_c[2i+1]$ for each LSB pair. Denoting the 7 most significant bits of an integer, a, as $MSB_7(a)$, we have

$$\begin{aligned} P\left(\mathbf{c}_{emb} = 0 | \mathbf{c}_{inv} = MSB_7(2i)\right) &= \frac{\mathbf{T}_c[2i]}{\mathbf{T}_c[2i] + \mathbf{T}_c[2i+1]} \\ &= 1 - P\left(\mathbf{c}_{emb} = 1 | \mathbf{c}_{inv} = MSB_7(2i)\right). \end{aligned} \tag{12.9}$$

The sender now collects all $\mathbf{S}[2i]$ DCT coefficients for the selected DCT frequency that are equal to $2i$ or $2i + 1$. Their LSBs are replaced with a segment of the message that was decompressed using an arithmetic decompressor to the length $\mathbf{S}[2i]$. The purpose of the decompressor is to change a uniformly distributed bitstream to a biased bitstream according to the probabilities given by Equation 12.9.

The recipient repeats the same steps that the sender followed, building a model for each DCT coefficient, and obtaining the probabilities given by Equation 12.9. Then, the bits are extracted from the LSBs and passed through an arithmetic compressor to obtain the secret message. This process is shown in Figure 12.4.

This example of model-based steganography preserves the *model* of the histograms of all individual DCT coefficients while achieving a relatively large embedding capacity (0.8 bits per nonzero DCT coefficient for 80% quality JPEG images).

We now calculate the average number of bits embedded per unit distortion for this algorithm (its embedding efficiency). Let $p_0 = P\left(\mathrm{c}_{emb} = 0 | \mathrm{c}_{inv} = MSB_7(2i)\right)$. The number of bits embedded at each DCT coefficient whose 7 MSBs are equal to $2i$ is $H(p_0) = -p_0 \log_2 p_0 - (1 - p_0)\log_2(1 - p_0)$. The probability of an embedding change is the probability that the LSB is different than the embedded message bit. Because the message bits are prebiased to have the probability of zero, p_0, we

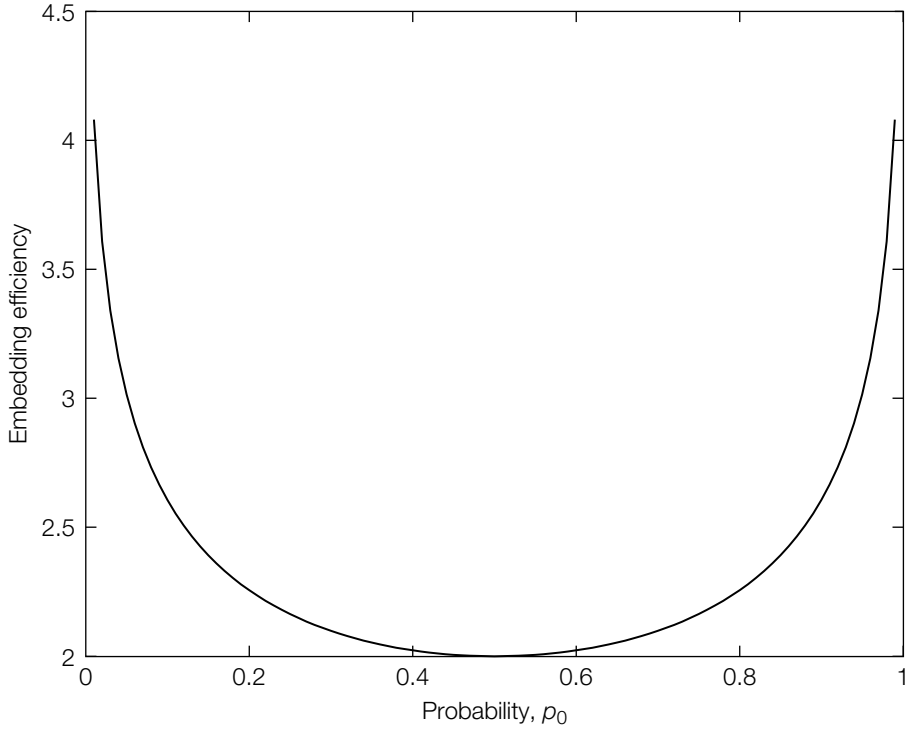


FIGURE 12.5

Embedding efficiency of model-based steganography.

have for the probability of change $p_0(1-p_0)+(1-p_0)p_0 = 2p_0(1-p_0)$. Thus, the embedding efficiency when embedding in coefficients with $MSB_7(2i)$ is

$$\frac{-p_0 \log_2 p_0 - (1-p_0)\log_2(1-p_0)}{2p_0(1-p_0)}. \tag{12.10}$$

The embedding efficiency is displayed in Figure 12.5. Note that is always greater than or equal to 2. This should be compared with embedding based on simple replacement of the LSBs. The latter's embedding efficiency is always equal to 2 because, on average, every other LSB is modified. In general, steganographic schemes with low embedding efficiency appear to be more statistically detectable than schemes with higher embedding efficiency. More detailed discussion on this topic appears in Section 12.5.1.

A more advanced version of this algorithm that preserves a specific second-order statistic was proposed in Sallee [357].

12.4.3 **Masking Embedding as Natural Processing**

In the previous sections, we discussed approaches to constructing practical steganographic schemes that strive to preserve some vital statistics (or a model) of Works. The advantage of such schemes is that they are secure in Cachin's sense as long as the statistics or model completely characterize the space of all Works. With the advancement of statistical image modeling, they also provide clues for building future schemes. Unfortunately, the job of the warden (steganalyst) is significantly simpler. Unlike Alice, the warden does not need to perfect her model of cover Works. All she needs to do is to find a deviation of Alice's model from reality. That could be a single statistical quantity that is not preserved by the embedder. It turns out that for each particular steganographic method, it is not that difficult to identify such a quantity. We give some examples in Section 13.2 on steganalysis. Unfortunately for Alice, while it is feasible to construct a steganographic scheme that preseves a specific model or statistics, it is much more complicated to preserve potentially hundreds of quantities used in current blind steganalysis methods.

This complication justifies investigation of alternative approaches to steganography, perhaps based on some heuristic principles. One possibility is to attempt to mask embedding as a natural process. Presumably, if the effect of embedding mimicked some natural process, it should be hard to argue that the Work was embedded rather than processed. In this section, we take a look at two practical realizations of this principle—one for images in the spatial domain and one for the JPEG format.

Stochastic Modulation

There are many noise sources that affect the acquisition of digital images. They include the shot noise due to quantum properties of light, dark current (the signal produced by the sensor when it is not exposed to light), circuit noise, readout noise, pixel-to-pixel nonuniformity (also known as pattern noise), quantization noise, etc. [181, 196, 207]. These stochastic components depend on ambient temperature and exposure time, or are intrinsic properties of the imaging sensor and the associated hardware.

The idea of stochastic modulation is to masquerade the embedding changes as a device noise. This is motivated by the plausible assumption that it may be difficult to determine whether the slightly increased noise level of the cover image is due to the presence of a hidden message or simply to environmental factors or different imaging hardware.

Let us assume that we want the impact of embedding to be the same as adding to the cover a noise signal with a given probability density function. Let $\mu[i]$ be an i.i.d. noise with pdf f_μ, and let $\mathbf{u}[i] = round(\mu[i])$

be the same sequence rounded to integers.[4] Thus, for any integers, k and i,

$$P(\mathbf{u}[i] = k) = \int_{k-0.5}^{k+0.5} f_{\mu}(x)dx. \tag{12.11}$$

The rounded noise sequence, $\mathbf{u}[i]$, is called the stego noise. One possibility to realize the embedding would be to use spread spectrum watermarking and embed 1 bit by adding to the cover image a spreading sequence modulated with the message bit. In order to reliably extract the message bits, however, the spreading sequence would have to be sufficiently long. This is not desirable for two reasons: we would impose a severe limit on the embedding capacity and also hide with a very low embedding efficiency (large distortion per embedded bit).

Before introducing stochastic modulation, we map the message bits to the interval $\{-1, 1\}$, rather than $\{0, 1\}$. We define a bit-assignment function, β, with the following property:

$$\beta(a + b, b) = -\beta(a, b) \tag{12.12}$$

for all integers a and b. The following function, for example, satisfies this requirement:

$$\begin{aligned} &\beta(a, b) = (-1)^a \quad \text{for} \quad 2i|b| \le a \le 2i|b| + |b| - 1, \\ &\beta(a, b) = -(-1)^a \quad \text{for} \quad (2i - 1)|b| \le a \le 2i|b| - 1 \\ &\beta(a, b) = 0 \text{ for } b = 0. \end{aligned} \tag{12.13}$$

The data embedding proceeds in the following manner. Using the stego key, the sender generates a pseudo-random path through the image and two independent stego noise sequences, $\mathbf{u}[i]$ and $\mathbf{v}[i]$, with the same probability mass function of Equation 12.11. The sender embeds the message bit, m, at the ith pixel, $\mathbf{c}[i]$, if and only if $\mathbf{u}[i] \neq \mathbf{v}[i]$. In this case, if $\beta(\mathbf{c}[i] + \mathbf{v}[i], \mathbf{u}[i] - \mathbf{v}[i]) = m$, the sender replaces $\mathbf{c}[i]$ with $\mathbf{s}[i] = \mathbf{c}[i] + \mathbf{v}[i]$. If $\beta(\mathbf{c}[i] + \mathbf{v}[i], \mathbf{u}[i] - \mathbf{v}[i]) = -m$, then due to the antisymmetry of Equation 12.12, we must have $\beta(\mathbf{c}[i] + \mathbf{u}[i], \mathbf{u}[i] - \mathbf{v}[i]) = -\beta(\mathbf{c}[i] + \mathbf{v}[i] + \mathbf{u}[i] - \mathbf{v}[i], \mathbf{u}[i] - \mathbf{v}[i]) = m$, and thus, we replace $\mathbf{c}[i]$ with $\mathbf{s}[i] = \mathbf{c}[i] + \mathbf{u}[i]$. If $\mathbf{u}[i] = \mathbf{v}[i]$, we replace $\mathbf{c}[i]$ with $\mathbf{s}[i] = \mathbf{c}[i] + \mathbf{u}[i]$ and reembed the same bit m at the next pixel along the pseudo-random walk.

The sender generates two independent stego noise sequences with the required probability mass function and then adds one or the other to embed the required bit at each pixel. If the two stego noise samples are the same, the embedding process cannot embed a bit, in which case the sender still adds

[4] Here, we again assume that the dynamic range of the cover Work is $0, \ldots, 255$.

the noise to the image but reembeds the same bit at the next pixel. Also, because images have finite dynamic range (e.g., the pixel values of a grayscale image are between 0 and 255), care must be taken when the embedding value, $\mathbf{s}[i] = \mathbf{c}[i] + \mathbf{v}[i]$, goes out of range. In this case, we must deviate from the stego noise model and instead add $\mathbf{v}'[i]$, such that $\beta(\mathbf{c}[i] + \mathbf{v}'[i], \mathbf{u}[i] - \mathbf{v}[i]) = m$ with $|\mathbf{v}'[i] - \mathbf{v}[i]|$ as small as possible.

To extract the message, the recipient first uses his or her stego key to generate both sequences, $\mathbf{u}[i]$ and $\mathbf{v}[i]$, and then reads the message as $\beta(\mathbf{s}[i], \mathbf{u}[i] - \mathbf{v}[i]\,|)$ skipping the cases when $\mathbf{u}[i] = \mathbf{v}[i]$.

INVESTIGATION

Embedding Capacity of Stochastic Modulation

The embedding capacity of stochastic modulation is $1 - P(\mathbf{u} = \mathbf{v}) = 1 - \sum_k [P(\mathbf{u}[i] = k)]^2$ bits per pixel and can be evaluated using Equation 12.11. For example, if μ is Gaussian $N(0, \sigma^2)$, the relative embedding capacity is shown in Figure 12.6. Note that by adding quantized Gaussian noise $N(0,1)$ the sender can be communicating at a little over 0.7 bits per pixel.

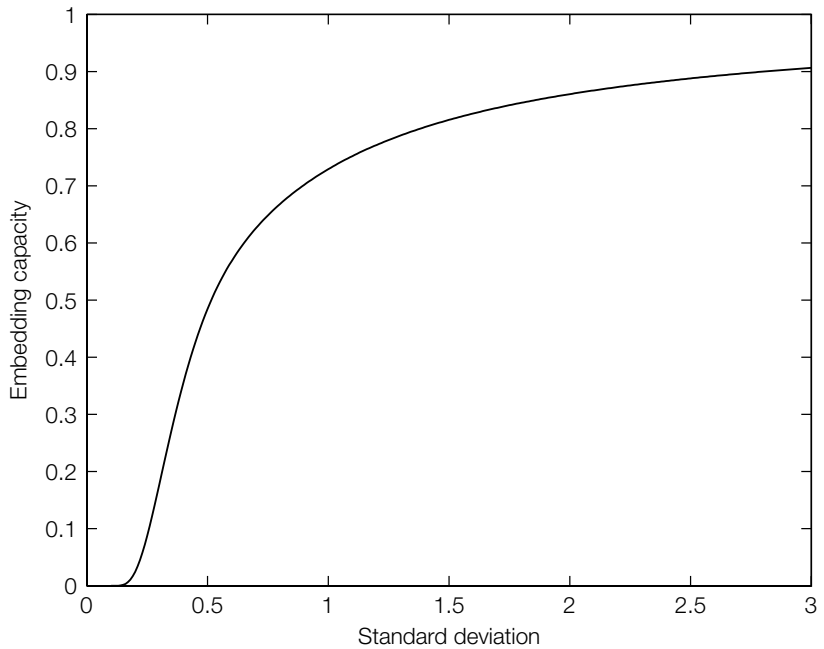


FIGURE 12.6

Embedding capacity of stochastic modulation.

It should not be surprising that the capacity is a function of the standard deviation, σ, because the noise sequences when rounded to integers are more likely to be equal for small σ.

Although stochastic modulation is much less detectable than LSB embedding, modern steganalysis tools based on feature extraction and classification (see Section 13.2.4 on blind steganalysis) are capable of detecting it relatively reliably, at least for payloads above 0.1 bpp. This is because the processing that occurs in digital cameras or scanners contains color interpolation and various filtering operations that create many complex dependencies among neighboring pixels. Injecting noise after all this processing disturbs these dependencies and enables relatively reliable steganalysis.

The F5 Embedding Algorithm

The F5 algorithm [442] was proposed with the goal to develop concepts and a practical embedding method for JPEG images that would provide high steganographic capacity without sacrificing security. Instead of LSB flipping, the embedding operation in F5 decrements the absolute value of the DCT coefficient by one. This operation preserves the natural shape of the DCT histogram, which looks after embedding as if the cover image was originally compressed using a lower quality factor.

The F5 algorithm embeds message bits along a pseudo-random path determined from a user pass-phrase. The DC terms and zero magnitude coefficients are skipped and not used for embedding.

When a coefficient's magnitude is changed from either 1 or −1 to zero, it is called "shrinkage." If shrinkage occurs, we have to reembed the same bit, which is always a 0, at the next coefficient. This is because the recipient will read the message bits from AC coefficients that are not equal to zero. However, if we reembed 0 bits, we will embed a *biased* bitstream with more zeros than ones. This would cause "staircase" artifacts in the histogram because of its monotonicity on $(-\infty, 0]$ and $[0, \infty)$. F5 solves this problem by redefining the LSB for negative numbers: $LSB(x) = 1 - x \mod 2$ for $x < 0$ and $LSB(x) = x \,\%\, 2$ otherwise. This solves the problem because now shrinkage occurs when embedding both 0s and 1s with approximately equal probability. This is due to the symmetrical shape of the DCT histogram.

The embedding capacity of F5 is $n - \mathbf{T}_c[0] - n/64 - (\mathbf{T}_c[-1] + \mathbf{T}_c[1])/2$, where n is the total number of DCT coefficients and $\mathbf{T}_c$ is the histogram of all DCT coefficients. The first three terms $n - \mathbf{T}_c[0] - n/64$ give the number of all nonzero AC coefficients, while the last term is the loss due to shrinkage. F5 is a relatively high-capacity algorithm that, on average, enables embedding

at 0.75 bits per nonzero DCT coefficient for JPEG images compressed with a quality factor of 80. As an additional improvement, for short messages F5 employs matrix embedding that increases the embedding efficiency. This general method for minimizing the number of embedding changes is explained in Section 12.5.1.

While the F5 algorithm modifies the histogram of DCT coefficients, some crucial characteristics of the histogram are preserved, such as its monotonicity and monotonicity of increments. The F5 algorithm cannot be detected using the histogram attack described in Section 13.2.1, because the embedding is not based on exchanging any fixed pairs of values. It is, however, vulnerable to attacks that use a process called calibration (see Section 13.2.3).

12.5 MINIMIZING THE EMBEDDING IMPACT

Cachin's definition of steganographic security calls for the Kullback-Leibler distance between the probability distributions of cover and stego Works to be as small as possible. Intuitively, we can try to achieve this by minimizing the distortion between the cover and stego Works and by restricting distortions to portions of the cover Work that are difficult to model. By so doing, we make it harder for the warden to collect evidence that steganography is taking place. This section addresses both issues (i.e., minimizing the embedding distortion and adaptive embedding).

We start by casting the problem of data hiding within the framework of coding theory. This enables us to derive bounds on the maximum number of secret message bits that can be embedded in a cover Work given a limited number of embedding changes. Remember that for LSB embedding, on average we embed 2 bits for each embedding change. This is because half the time, the LSB value is the same as the message bit and therefore no change is needed. In fact, we can do much better than this if the message is shorter than the embedding capacity. Section 12.5.1 discusses this in detail.

Section 12.5.2 then discusses methods by which the embedder can adaptively determine the location of the alterations. A shared selection rule is no longer required, that is, Bob does not need to know the location of the changes in order to decode the message. The advantage of this is that Alice can now utilize side information only available to her in order to minimize the impact of embedding. For example, assuming she has access to *unquantized* DCT coefficients while JPEG compressing the cover Work, she can confine her embedding changes to those coefficients that experience the smallest *combined* distortion due to quantization and embedding.

12.5.1 Matrix Embedding

The embedding changes can encode the message bits in many different ways. For example, in LSB embedding, the LSBs are changed to match the secret message bits (e.g., the extracted message is simply the string of LSBs of individual pixels). This way, we always embed, on average, 2 bits per one embedding change. However, we can do substantially better if we adopt a more clever embedding scheme. In particular, if the message is shorter than the embedding capacity, one can, for example, use the position of changes to encode more bits per one change. Let us take a look at the following simple example.

Let us assume that we have a group of three pixels with grayscale values x_1, x_2, x_3, and we wish to embed 2 message bits, b_1, b_2. It seems that a reasonable approach might be to simply embed b_1 at x_1 and b_2 at x_2, modifying the LSB of the pixels to match the corresponding message bits. Assuming the 2 bits are 0 or 1 with equal probability, the expected number of changes to the whole group of pixels to embed both bits is 1. Thus, we embed at embedding efficiency of 2 or 2 bits per one change. However, we can do better. Let us encode $b_1 = LSB(x_1)$ XOR $LSB(x_2)$ and $b_2 = LSB(x_2)$ XOR $LSB(x_3)$. If the values of the cover Work satisfy both equations with equality, no embedding changes are necessary. If the first one is satisfied but not the second one, we simply flip the LSB of x_3. If the second one is satisfied but not the first one, we flip the LSB of x_1. If neither one is satisfied, we flip x_2. Because all four cases are equally likely with probability 1/4, the expected number of changes is 3/4, which is less than what we had before. This simple trick is a special case of a much more general principle called *matrix embedding*, [40, 159, 419], which we now describe in more detail.

We will assume that the individual elements of the cover Work are represented as bits via some bit-assignment function, β (e.g., the LSB). The exact mechanism of the embedding operation is not essential to us now. We only need to know that the operation changes the bit assigned to the element.

Let us assume that we have a pair of embedding and extraction functions such that we can embed any message $\mathbf{m} \in \mathcal{M}$ using, at most, R changes

$$Emb\colon \{0,1\}^n \mathcal{M} \to \{0,1\}^n \text{ and } Ext\colon \{0,1\}^n \to \mathcal{M}, \tag{12.14}$$

such that for all $\mathbf{m} \in \mathcal{M}$ and $\mathbf{x} \in \{0,1\}^n$,

$$Ext(Emb(\mathbf{x}, \mathbf{m})) = \mathbf{m}$$
$$d(\mathbf{x}, Emb(\mathbf{x}, \mathbf{m})) \leq R.$$

As defined in Section 12.2, the embedding capacity in bits is $\log_2 |\mathcal{M}|$. If R_a is the expected number of embedding changes over uniformly distributed cover Works, $\mathbf{x}$, and messages, $\mathbf{m}$, then $R_a \leq R$, and we define the *embedding efficiency*, of this scheme as $e = \frac{\log_2 |\mathcal{M}|}{R_a}$. We also define the *lower embedding*

efficiency, $\underline{e} = \frac{\log_2 |\mathcal{M}|}{R}$, which is always less than or equal to the embedding efficiency, e, since $R_a \leq R$.

Matrix embedding is a form of syndrome coding. To define a syndrome, remember that for an $[n, k]$ linear code, $\mathcal{C}$, k-bits are transmitted as an n-bit code word, where $n > k$. In error correcting codes, the k-bits are usually referred to as the message, but for reasons that will soon be apparent, we refer to them as the base bits. Each $[n, k]$ linear code, $\mathcal{C}$, is completely described by its generator matrix, $\mathbf{G}$, which is a regular binary matrix with k rows and n columns. All code words, $c \in \mathcal{C}$, are obtained as linear combinations of the rows of $\mathbf{G}$, where the coefficients in the linear combination are the k base bits.

The detector receives a possibly corrupted code word, $\mathbf{c}'$. The vector, $\mathbf{Hc}'$, is called the syndrome, where $\mathbf{H}$ is the $(n-k) \times n$ parity check matrix of the code. If $\mathbf{c}'$ is an element of $\mathcal{C}$ (i.e., is uncorrupted), then its syndrome is

$$\mathbf{Hc}' = 0. \tag{12.15}$$

If $\mathbf{c}'$ is corrupted, then

$$\mathbf{Hc}' \neq 0.$$

Note that the syndrome is an $(n-k)$ dimensional vector.

As discussed in Appendix A.2, there are many corrupted code words that map to the same syndrome, $\mathbf{s}$, and this set of corrupted code words forms a coset that we denote as $\mathcal{C}(\mathbf{s})$ (i.e., the set of $\mathbf{c}'$), such that $\mathbf{Hc}' = \mathbf{s}$.

Matrix embedding (and syndrome coding) hide the steganographic message as the syndrome. Thus, for an $[n, k]$ code, we can embed $(n-k)$ steganographic message bits. Since the message is encoded in the syndrome and *not* in the k base bits, as is the case for conventional coding for error correction, we refrain from calling the k base bits message bits.

To perform syndrome coding, let $\mathbf{m}$ denote our $(n-k)$-bit message (syndrome). To encode $\mathbf{m}$, first, let $\mathbf{x}$ denote the n-bits from the cover Work that will hide the message. We must alter $\mathbf{x}$ to $\mathbf{y}$ such that

$$\mathbf{Hy} = \mathbf{m}.$$

In other words, the recipient extracts the message from the stego Work $\mathbf{y}$ as its syndrome. Let $\mathbf{y} = \mathbf{x} + \mathbf{e}$, where $\mathbf{e}$ is the vector of embedding changes. The Hamming weight of $\mathbf{e}$ is equal to the number of embedding changes. Then, we have that

$$\mathbf{H}(\mathbf{x} + \mathbf{e}) = \mathbf{Hy} = \mathbf{m},$$

and thus,

$$\mathbf{He} = \mathbf{m} - \mathbf{Hx}. \tag{12.16}$$

The right-hand side of Equation 12.16 defines a syndrome coset,[5] $\mathcal{C}(\mathbf{m} - \mathbf{Hx})$, and the solution to Equation 12.16 is that $\mathbf{e}$ is the coset leader, $\mathbf{e_L}$, of $\mathcal{C}(\mathbf{m} - \mathbf{Hx})$, that is, the element of the coset with the smallest Hamming weight (smallest number of embedding changes). And, since the Hamming weight of $\mathbf{e_L}$ is less than or equal to the covering radius, R, of the code, then the number of embedding changes is also less than or equal to R. We can summarize by stating that a linear $[n, k]$ code, $\mathcal{C}$, with covering radius, R, can be used to construct an embedding scheme capable of communicating $(n - k)$ bits using at most R changes.

Depending on the choice of the linear code, C, different matrix embedding schemes can be constructed. The simplest scheme that is, for instance, implemented in the steganographic algorithm F5 [442] is based on binary Hamming codes with $n = 2^p - 1$ and $k = 2^p - 1 - p$, where p is a positive integer. The parity check matrix, $\mathbf{H}$, of this code has dimensions $p \times (2^p - 1)$, and its columns are binary representations of the numbers $1, \ldots, 2^p - 1$. We can say that $\mathbf{H}$ has as its columns all possible nonzero syndromes. For example, the parity check matrix, $\mathbf{H}$, for a Hamming code with $p = 3$ is

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Notice that any nonzero syndrome, $\mathbf{s}$, is a column in $\mathbf{H}$, say the ith column. The coset leader, $\mathbf{e_L}$, of $\mathcal{C}(\mathbf{s})$ is a vector with only a nonzero element in its ith position. Thus, all coset leaders of this code have Hamming weight 1 and a code covering radius $R = 1$.

To summarize, Hamming codes enable embedding of $n - k = p$ bits in $n = 2^p - 1$ pixels by making at most one change. The reader is encouraged to verify that the "trick" shown in the introduction of this section corresponds to matrix embedding using Hamming codes with $p = 2$ and parity check matrix

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}.$$

INVESTIGATION

Matrix Embedding Using Hamming Codes

In this investigation, we explain how matrix embedding using Hamming codes can be implemented in practice. The benefit of using this coding scheme is

[5] Note that the sender knows the syndrome, $\mathbf{s}$, because $\mathbf{m}$ is the message to be communicated and $\mathbf{x}$ is the vector of cover Work bits.

demonstrated by showing the decrease in the average number of embedding changes needed to embed a message.

Let us assume that the sender wants to communicate K bits in a cover Work consisting of N pixels. This means that the relative message length is $\alpha = K/N$. Since matrix embedding using Hamming codes embeds p bits in $2^p - 1$ pixels, the sender will minimize the number of embedding changes by choosing the largest value of p that provides sufficient relative payload, that is,

$$\alpha_{p+1} = \frac{p+1}{2^{p+1}-1} < \alpha \leq \frac{p}{2^p - 1} = \alpha_p.$$

The sender starts by dividing the cover into $N/(2^p - 1)$ blocks, each consisting of $2^p - 1$ pixels. In each block, p message bits are embedded by performing at most one embedding change in the same manner as above, communicating p bits as a syndrome $\mathbf{m} = \mathbf{Hy}$, where $\mathbf{y}$ is the vector of bits from the stego image.

We now calculate the average number of embedding changes and the embedding efficiency of this matrix embedding scheme. Remember that the embedding efficiency is the average number of message bits embedded per unit distortion (one embedding change).

When embedding p pseudo-random bits, $\mathbf{m}$, in a pixel block, then, with probability $1/2^p$, the cover pixel block, $\mathbf{x}$, will already communicate the required message (i.e., $\mathbf{m} = \mathbf{Hx}$), and no embedding change will be necessary in this case. In all other cases, the sender makes exactly one embedding change with probability $1 - 1/2^p$. Thus, the average number of embedding changes to embed p bits is

$$0 \times 1/2^p + 1 \times (1 - 1/2^p) = 1 - /2^p,$$

and the embedding efficiency, e_p, is

$$e_p = \frac{p}{1 - 2^{-p}}, \tag{12.17}$$

Table 12.1 shows the relative message length and the embedding efficiency for various values of p. Note that with increasing p, the embedding efficiency increases while the relative payload decreases. Also, Hamming codes cannot be applied for embedding messages of relative length larger than 2/3.

Finally, note that the parameter, p, chosen by the sender depends on the message length and thus needs to be communicated to the recipient in the stego image itself. This can be arranged in a variety of ways, depending on the steganographic scheme. For example, the sender can reserve a few pixels in the image and embed information about p using LSB embedding.

Table 12.1 Relative message length and embedding efficiency for matrix embedding using binary Hamming codes $[2^p - 1, 2^p - 1 - p]$.

p	Relative message length, α_p	Embedding efficiency, e_p
1	1.000	2.000
2	0.667	2.667
3	0.429	3.429
4	0.267	4.267
5	0.161	5.161
6	0.093	6.093
7	0.055	7.055
8	0.031	8.031
9	0.018	9.018

Upper Bound on Embedding Efficiency

In the previous section, we saw how linear codes can be used to decrease the number of changes needed to embed a message using a procedure called matrix embedding. A natural question to ask is, what are the limits of this approach? For example, given a relative message length, α, what is the average minimal number of embedding changes needed to embed it? Alternatively, we may ask about the maximum relative message length embeddable using a given number of changes. Answering these questions is the subject of this section. The bounds derived here serve as a benchmark against which matrix embedding schemes can be evaluated.

The maximum number of messages that can be communicated by making at most R changes is bound by the total number of ways one can make R or fewer changes. This is because the sender and the receiver could share a lookup table where each embedding change would be associated with one message. Because there are $\sum_{i=0}^{R} \binom{n}{i} 2^i$ ways in which one can make R or fewer changes in n pixels, we obtain the following bound on the maximal number of bits that can be embedded:

$$\log_2 |\mathcal{M}| \leq \log_2 \sum_{i=0}^{R} \binom{n}{i} 2^i. \tag{12.18}$$

We now apply a well-known inequality from information theory [89]:

$$\log_2 \sum_{i=0}^{R} \binom{n}{i} 2^i \leq nH(R/n), \tag{12.19}$$

where $H(x) = -x\log_2 x - (1-x)\log_2(1-x)$ is the *binary entropy function* shown in Figure 12.7. Combining Equations 12.18 and 12.19, we obtain an upper bound on the maximal relative payload embeddable using R changes, that is

$$\alpha_{max} = \frac{\log_2 |\mathcal{M}|}{n} \leq H(R/n). \tag{12.20}$$

Sometimes, we may be interested in finding the maximal possible embedding efficiency *given* a certain relative payload α. This would answer the question of what is the minimal number of embedding changes that are, on average, needed to embed a given payload. This bound can be obtained by first rewriting

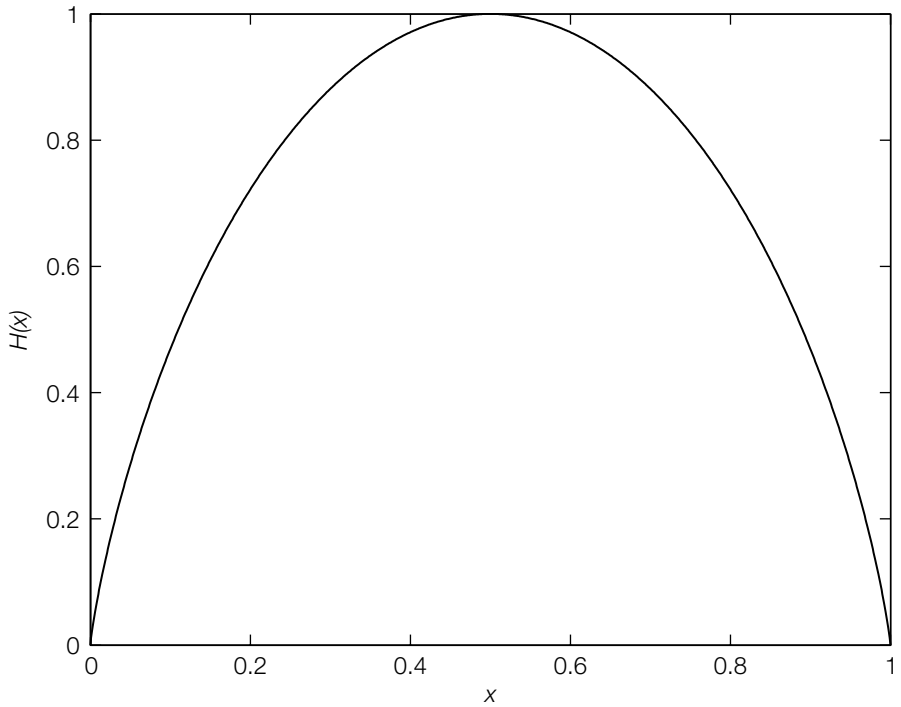


FIGURE 12.7

Binary entropy function.

Equation 12.20 as $n/R \leq 1/H^{-1}(\alpha)$, or $\log_2 |\mathcal{M}|/R \times n/\log_2 |\mathcal{M}| \leq 1/H^{-1}(\alpha)$, which gives

$$\underline{e} = \frac{\log_2 |\mathcal{M}|}{R} \leq \frac{\alpha}{H^{-1}(\alpha)}, \tag{12.21}$$

where the range of the inverse function H^{-1} is the interval $[0, 1/2]$. This last bound gives a useful bound on the lower embedding efficiency achievable using an arbitrary matrix embedding scheme that can embed a relative message length, α. We note that this bound is tight and can be reached with matrix embedding [159]. We plot the bound as a function of the relative message length, α, in Figure 12.8. The naive LSB embedding method always performs with an embedding efficiency $e = 2$ at any payload, α. Binary Hamming codes, also shown in the figure, provide substantial advantage, for example, giving an embedding efficiency as high as $e = 6$ for a relative message length $\alpha = 0.1$.

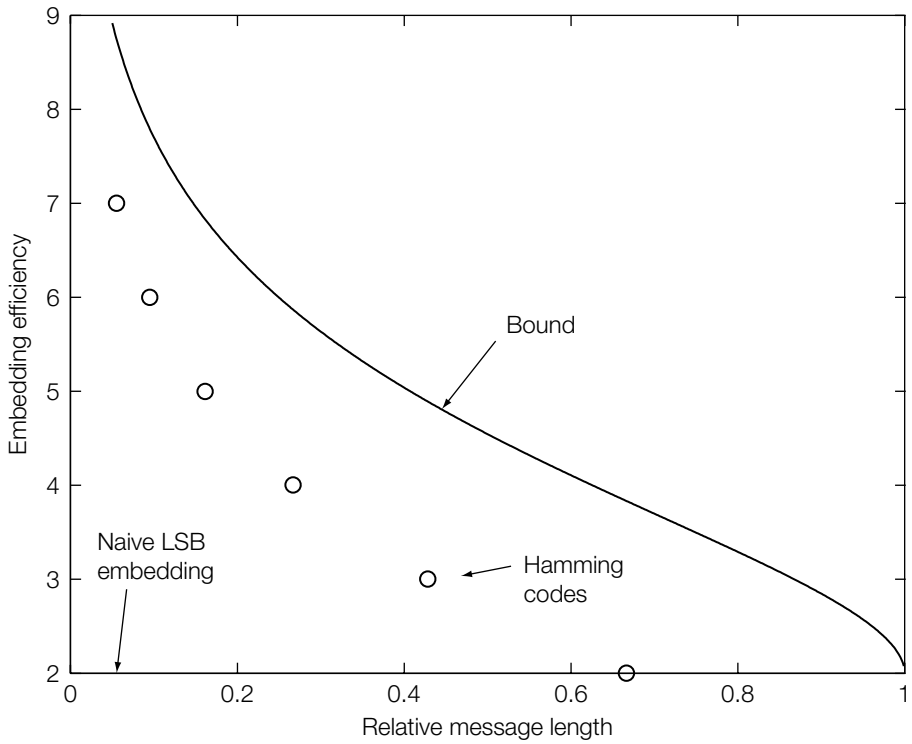


FIGURE 12.8

Embedding efficiency, e_p, in bits per embedding change as a function of relative message length, α, for matrix embedding using binary Hamming codes.

In general, it is fairly difficult to find codes with embedding efficiency close to the bound of Equation 12.21. Random linear codes with small codimension ($n - k$) have been shown to achieve slightly better performance than Hamming codes [149]. Nonlinear codes offer better embedding efficiency at the cost of a more complicated embedding [39]. Matrix embedding methods for large payloads were studied in Fridrich and Soukal [152].

12.5.2 Nonshared Selection Rule

In the previous section, we introduced matrix embedding as a means for decreasing the number of embedding changes and thus reducing the statistical impact of embedding a message into a cover Work. Besides making fewer embedding changes, the sender may attempt to restrict the embedding to those parts of the cover Work where it is intuitively more difficult to detect the changes. The rule used to select the placement of embedding changes is called the *selection rule*. If the selection rule is only known to the sender but not to the recipient, we speak of a *nonshared* selection rule.

To motivate the need for nonshared selection rules in steganography, imagine the following situation. The sender has a raw, never compressed image and wants to embed information in its JPEG compressed form. Can the knowledge of the raw image help us better conceal the embedding changes? Intuitively, it should. For example, when compressing the image, the sender can inspect the DCT coefficients after they are divided by the associated quantization steps but *before* they are rounded to integers, and select for embedding those coefficients whose fractional part is close to 0.5. Such coefficients experience the *largest* quantization error during JPEG compression and the *smallest combined error* (rounding + embedding) if rounded to the other value. When rounding the coefficient 5.47, for instance, we can embed a bit by rounding it to 5 or to 6. The rounding distortion (without embedding) is 0.47. If embedding requires rounding to 6, the combined rounding and embedding distortion is only slightly larger, 0.53. The obvious problem with this idea, however, is that the recipient will not be able to tell which DCT coefficients in the stego JPEG file were used for embedding, because it is not possible to completely undo the loss due to rounding by JPEG compression. This is an example of a situation in which the sender *cannot* share the selection rule with the recipient.

The problem of nonshared or partially shared selection rules occurs quite frequently in steganography. Consider, for example, adaptive steganography [71, 220] where the pixels in the cover Work are chosen using a selection rule based on their neighborhood. Because the act of embedding itself modifies the pixel values, then, depending on the selection rule, it is possible that the recipient will not recover the same set of message-carrying pixels as the sender. As an example, consider the following selection rule. The sender calculates for each pixel in the cover Work the variance of all pixels in its local 3×3 neighborhood and embeds m message bits using LSB embedding into the LSBs of m pixels

with the largest local variance. When the same selection rule is applied by the recipient of the stego image, it may well happen that the m pixels with the largest local variance will not be completely the same as those selected by the sender. This may prevent the recipient from extracting the message. It obviously would be very useful to have a general method that would enable construction of steganographic schemes with nonshared selection rules.

Communication using nonshared selection rules in steganography has been called "writing on wet paper" [150]. To explain this metaphor, imagine that the cover image, $\mathbf{x}$, was exposed to rain and the sender is only allowed to slightly modify the dry spots of $\mathbf{x}$ (the selected pixels) but not the wet spots. During transmission, the stego image, $\mathbf{y}$, dries out and thus the recipient does not know which pixels the sender used (the recipient has no information about the dry pixels). This is equivalent to a well-known problem in information theory called *writing in memory with defective cells* [119, 182, 412, 466]. Here, a computer memory contains n cells, out of which $(n - k)$ cells are permanently stuck at either 0 or 1. The device that writes data into the memory knows the locations and status of the stuck cells. The task is to write as many bits as possible into the memory (up to k) so that the reading device, which does not have any information about the stuck cells, can correctly read the data. Clearly, writing on wet paper is formally equivalent to writing in memories with stuck cells (stuck cells = wet pixels).

The defective memory is a special case of the Gel'fand-Pinsker channel with an informed sender [161] that we already encountered in Chapter 5. It turns out that syndrome coding, which we discussed in the previous section on matrix embedding, is an indispensible tool for solving the problem of nonshared selection rules.

Wet Paper Codes with Syndrome Coding

In this section, we present a coding method by which it is possible to construct steganographic schemes with nonshared selection rules. The method is similar to matrix embedding as it is also based on syndrome codes (i.e., the message is communicated as a syndrome of some linear code).

We again assume that the Work is represented via some bit-assigment function, as a vector of bits, $\mathbf{x}$, for example, the LSBs. The sender uses some selection rule to choose k changeable pixels $\mathbf{x}[j]$, $j \in \mathcal{J} \subset \{1, \ldots, n\}$, and $|\mathcal{J}| = k$. Using the writing on wet paper metaphor, the set of pixel indices $\mathcal{J}$ indicate "dry" pixels that may be modified by the sender. The remaining pixels, corresponding to "wet" pixels, are not to be modified during embedding.

Following the mechanism of syndrome codes, to communicate m bits $\mathbf{m} \in \{0, 1\}^m$, the sender modifies some changeable pixels in the cover Work so that the stego Work, $\mathbf{y}$, satisfies

$$\mathbf{D}\mathbf{y} = \mathbf{m}, \tag{12.22}$$

where $\mathbf{D}$ is an $m \times n$ parity check matrix shared by the sender and the recipient. This embedding mechanism is almost identical to matrix embedding with one important difference: The sender now cannot modify all pixels but only the changeable ones. The recipient, again, extracts the message by calculating the syndrome, $\mathbf{Dy}$, from the stego Work, $\mathbf{y}$.

The embedding requires solving the system of linear equations of Equation 12.22, where the unknowns are the bits, $\mathbf{y}[j], j \in \mathcal{J}$, and the remaining bits of $\mathbf{y}$ are known. In order to better see what kind of task the sender has to perform, we rewrite Equation 12.22 as

$$\mathbf{Dv} = \mathbf{m} - \mathbf{Dx} \tag{12.23}$$

using the variable $\mathbf{v} = \mathbf{y} - \mathbf{x}$. The nonzero elements of $\mathbf{y}$ correspond to the pixels the sender must change to satisfy Equation 12.22. In Equation 12.23, there are k unknowns, $\mathbf{v}[j], j \in \mathcal{J}$, corresponding to changeable pixels, while the remaining $(n - k)$ values, $\mathbf{v}[i], i \notin \mathcal{J}$, are known and must be equal to zero. Thus, on the left-hand side, the sender can remove from $\mathbf{D}$ all $(n - k)$ columns corresponding to indices $i \notin \mathcal{J}$ and also remove from $\mathbf{v}$ all $(n - k)$ elements $\mathbf{v}[i]$ with $i \notin \mathcal{J}$. Keeping the same symbol for the pruned vector, $\mathbf{v}$, Equation 12.23 becomes

$$\mathbf{Hv} = \mathbf{z}, \tag{12.24}$$

where $\mathbf{H}$ is an $m \times k$ matrix consisting of those k columns of $\mathbf{D}$ corresponding to indices $\mathcal{J}$, $\mathbf{v}$ is an unknown $k \times 1$ vector holding the embedding changes, and $\mathbf{z} = \mathbf{m} - \mathbf{Dx}$ is a known $m \times 1$ right-hand side. At this point, the problem becomes *equivalent* to matrix embedding with a parity check matrix $\mathbf{H}$. Finding any member, $\mathbf{v}$, of the coset, $\mathcal{C}(\mathbf{z})$, amounts to successfully embedding the message. Choosing the coset leader minimizes the number of embedding changes.

When we carefully compare this task with matrix embedding, we discover one important difference. While in matrix embedding, we could impose a specific structure on the matrix, $\mathbf{H}$, such as requiring it to be the parity check matrix of Hamming codes; here, $\mathbf{H}$ is obtained as a submatrix of a larger matrix, $\mathbf{D}$, via a selection process dictated by the cover Work or by some side information. Thus, when embedding into two cover Works of the same dimensions, the two submatrices will, in general, be different. It is not easy to impose a structure on $\mathbf{H}$ that would enable us to use some known codes.

One strategy the sender may accept is to simply solve Equation 12.24 using Gaussian elimination. However, the complexity of Gaussian elimination is cubic in the number of equations, k, which is the number of changeable pixels. Thus, using Gaussian elimination to embed large payloads would not be feasible. Another possibility is to use random, sparse matrices, $\mathbf{D}$, for which the task of solving the linear system can become very easy. This idea is explored in the next section.

Nonshared Selection Rules Using the Matrix LT Process

In the previous section, we explained how nonshared selection rules can be realized in steganography using syndrome coding and a parity check matrix, **D**, shared between the sender and the recipient. In this section, we show how solving the linear system of Equation 12.24 can be implemented very efficiently, by choosing **D** from a special class of random, sparse matrices.

To explain the basic idea, imagine that in the linear system $\mathbf{Hv} = \mathbf{z}$ that the sender needs to solve, there exists a column with exactly one 1 in, say, the $j(1)$th row. The sender swaps this column with the first column and then swaps the first and $j(1)$th rows to bring the 1 to the upper left corner of **H**. Now we will apply the same step again while ignoring the first column and row of the permuted matrix. We again find a column with only one 1, say, in the $j(2)$th row[6] and swap the column with the second column of **H** followed by swapping the second and $j(2)$th rows. As a result, we will obtain a matrix with one's on the first two elements of its main diagonal and zeros below them. We can continue this process, this time ignoring the first two columns and rows. If we are lucky, this process of row and column swapping will eventually produce a permuted matrix with an upper-diagonal form. Such a system can be efficiently solved using standard back substitution, as in Gaussian elimination. We call this permutation procedure the *matrix LT process* because it was invented in the theory of erasure correcting codes called LT codes [275]. Note that if, at some step during the permutation process, we cannot find a column with exactly one 1, then the matrix LT processed failed.

The big question is what properties should the matrix **D** have in order for the permutation process to successfully finish? It turns out that as long as the Hamming weights of columns of **D** (and thus of **H**, too) follow a certain probability distribution, then, with high probability, the matrix LT process will succeed. This distribution is known as a robust soliton distribution (RSD) [275] and is defined as follows.

The probability that a column in **D** has Hamming weight i, $1 \leq i \leq k$, is $\frac{\rho[i] + \tau[i]}{\eta}$, where

$$\begin{aligned}
\rho[i] &= \tfrac{1}{m} \quad i = 1 \\
&= \tfrac{1}{i(i-1)} \quad i = 2, \ldots, m, \\
\tau[i] &= \tfrac{R}{im} \quad i = 1, \ldots, m/R - 1 \quad \text{and} \\
&= \tfrac{R \ln(R/\delta)}{m} \quad i = m/R \\
&= 0 \quad i = m/R + 1, \ldots, m,
\end{aligned}$$

[6] Since we are ignoring the first row, this column may have another 1 as its first element.

$$\eta = \sum_{i=1}^{m} (\rho[i] + \tau[i]) \tag{12.25}$$

and

$$R = c \ln(m/\delta)\sqrt{m} \tag{12.26}$$

for some suitably chosen constants δ and c. The choice of these constants will be discussed later.

An example of an RSD distribution for $k = 1{,}000$, and $c = 0.1$ is shown in Figure 12.9. By inspecting the figure, we can get a feeling for why this distribution looks the way it does. First, note that the distribution prefers columns of low Hamming weight while denser columns are less frequent. This intuitively guarantees that the LT process will always find a column with just one 1. The purpose of the mysterious spike at Hamming weight 27 is to make sure that the matrix will be regular with high probability. Without the spike, the matrix would become too sparse and not be of full rank.

To generate a matrix where the number of ones in its columns follows an RSD distribution, we first generate a sequence of integers, $w_1, w_2, \ldots$, that

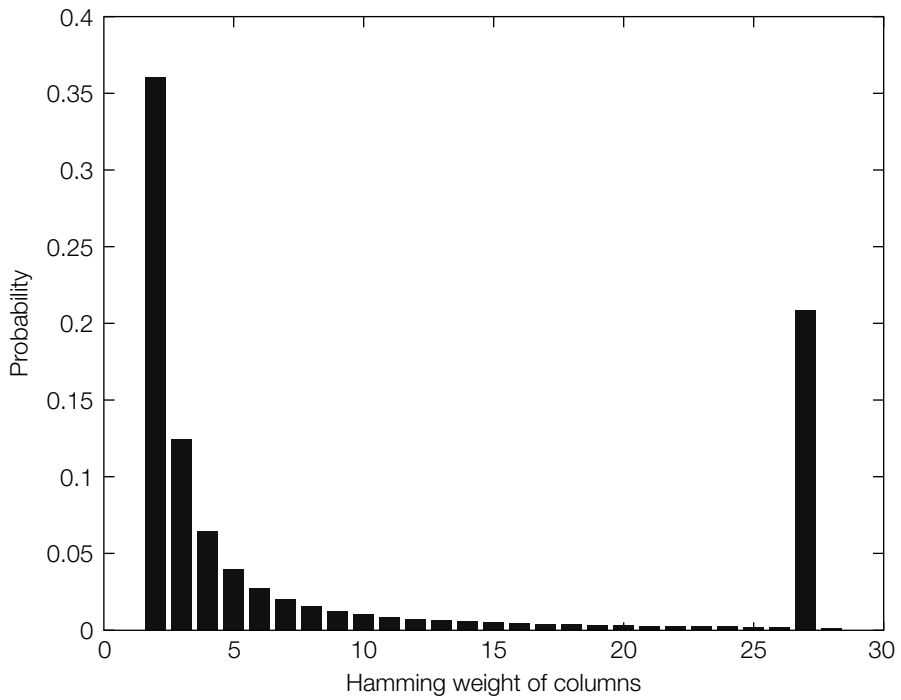


FIGURE 12.9

Robust soliton distribution for $k = 1{,}000$ and $c = 0.1$.

follows the distribution. Then, for each w_i, a column is generated by applying a random permutation to a column containing w_i ones and $k - w_i$ zeros.

It is known [275] that when the Hamming weights of the columns of **D** (and thus of **H**) follow the RSD distribution given by Equation 12.25, then the probability that the matrix LT process finishes successfully is greater than $1 - \delta$, provided that the message length, m, and the number of changeable pixels, k, satisfy the following inequality:

$$k > \theta m = m + O(\sqrt{m}\ln^2(m/\delta)). \tag{12.27}$$

The term $O(\sqrt{m}\ln^2(m/\delta))$ is the embedding capacity loss, since we are embedding m bits into $k > m$ changeable pixels. For $c = 0.1$, $\delta = 5$, and $m = 10{,}000$, the capacity loss is about 6% ($\theta = 1.062$) while the probability that the LT process succeeds is about 75%. This probability increases, and the capacity loss decreases, with increasing message length (see Table 12.2).

We now describe how this proposed coding can be incorporated into a steganographic method. The sender uses a secret stego key and forms the matrix **D** with columns following the RSD distribution. Then, he or she solves Equation 12.24 and finds the solution, **v**, using the Matrix LT procedure. The vector, **v**, tells the sender which pixels in the image should be modified.

The receiver forms the matrix **D** and extracts the message as $\mathbf{m} = \mathbf{Dy}$. However, in order to do so, the recipient needs to know the message length, m, because the RSD distribution, and thus **D**, depends on m. The remaining parameters c and δ can be publicly known.

We now discuss how to communicate the message length, m, to the receiver, together with other implementation details, such as what the sender should do if the matrix LT process fails. While there exist several different strategies, we describe the simplest one. The parameter, m, can be communicated simply by reserving a small portion of the cover image (e.g., determined from

Table 12.2 Running time (in seconds) for solving $m \times m$ and $m \times \theta m$ linear systems using Gaussian elimination and the matrix LT process, respectively ($c = 0.1$, $\delta = 5$); P is the probability of a successful pass through the LT process.

Runtime, overhead θ, and probability of successful pass				
m	**Gauss**	**LT**	θ	P
1,000	0.023	0.008	1.098	43%
10,000	17.4	0.177	1.062	75%
30,000	302	0.705	1.047	82%
100,000	9320	3.10	1.033	90%

the stego key) where we can use wet paper codes with a random matrix, **D**, that has a uniform distribution of 0s and 1s, and solve the system using Gaussian elimination. Since we only need, say, 20 bits to communicate m, solving a system of 20 equations using Gaussian elimination will be fast. The message itself is communicated in the remainder of the image using the matrix LT procedure whose matrix, **D**, follows the RSD distribution.

To deal with occasional failures of the matrix LT procedure, we can make **D** dependent on the message length, that is, make the seed for the pseudo-random number generator that is used to construct the RSD distribution be dependent on a combination of the stego key and message length. By doing so, we can simply add a dummy bit to the message, regenerate **D**, and repeat the embedding process until we obtain a successful pass through the matrix LT process.

The complexity of message embedding is $O(n\ln(k/\delta) + k\ln(k/\delta)) = O(n\ln(k/\delta))$ [150]. The first term arises from evaluating the product, **Dx**, while the second term is the complexity of the LT procedure. The gain in computational efficiency over regular Gaussian elimination, which has complexity $O(k^3)$, is substantial, as is demonstrated in the following Investigation and enumerated in Table 12.2.

INVESTIGATION

Matrix LT Process

In this investigation, we describe the LTSOLVER system that provides a fast solution to m linear equations, $\mathbf{Hv} = \mathbf{z}$, with k unknowns, $\theta m < k$, when the Hamming weights of the columns of **H** follow the RSD distribution of Equation 12.25.

System 20: SE_LTSOLVER

The function LTSOLVER accepts a binary $m \times k$ matrix **H** and a binary m-dimensional vector, **z**, as input and returns a solution, **v**, to the system using the Matrix LT procedure. It is based on the following algorithm:

0. $i = 0, t = 0$

1. WHILE $i < m$
{
$i = i + 1$

IF
there exists a column $i' \geq i$ with exactly one 1 in rows $j \geq i$.
THEN

swap the j-th and i-th rows and swap the corresponding
bits in the right hand side $\mathbf{z}[i]$ and $\mathbf{z}[j]$.
Also, swap the
i-th and i'-th columns and corresponding unknowns
$\mathbf{v}[i]$ and $\mathbf{v}[i']$.
Set $t = t + 1$ and store the transposition
$i \leftrightarrow i'$ as $\tau[t]$).

ELSE
declare a failure and STOP.

END WHILE

2. At this point, $\mathbf{H}$ has been permuted to the upper diagonal form.

Set $\mathbf{v}[i] = 0$ for $m < i \leq k$ and finish the solving
process using the usual back-substitution as in Gaussian elimination.

3. Apply the sequence of transpositions τ in the opposite order to $\mathbf{v}$,
$\mathbf{v} \leftarrow \tau[1](\tau[2](\ldots(\tau[t]\mathbf{v})\ldots))$.

4. The resulting $\mathbf{v}$ is the solution to $\mathbf{Hv} = \mathbf{z}$.

Experiment

The performance of the LTSOLVER was tested on matrices H of dimension $m \times \theta m$ whose column weights follow the RSD distribution with $c = 0.1$ and $\delta = 5$. Table 12.2 shows the running times for the Matrix LT process and the Gaussian elimination. It also shows the parameter, θ, and the probability of failure, P. The experiments were performed on a standard Pentium PC with a 3.4 MHz processor. The table clearly illustrates orders of magnitude reduction in computational time of the LT procedure over standard Gaussian elimination.

Perturbed Quantization

We now discuss perturbed quantization as a final example of steganographic algorithms based on nonshared selection rules. Interestingly, these techniques lead to some of the most secure steganographic schemes known today [148, 166].

We begin by generalizing the motivational example from Section 12.5.2 in which the sender embedded message bits during JPEG compression. The idea behind perturbed quantization is to couple the embedding procedure with common image-processing functions such as lossy compression, resampling, and filtering. This is accomplished by perturbing the final quantization step

associated with these operations. The sender's goal is to minimize the *combined* distortion due to embedding *and* image processing. This is an example of a steganographic algorithm with a nonshared selection rule, since the placement of embedding changes is determined by the original cover Work, which is a side information only available to the sender.

We first describe the general framework of perturbed quantization, followed by a heuristic security analysis. Let us assume that the cover Work is represented with a vector $\mathbf{x} \in \mathcal{P}^m$, where $\mathcal{P}$ is the range of its elements (pixels, coefficients, colors, indices). For example, for an 8-bit grayscale image, $\mathcal{P} = \{0, \ldots, 255\}$ and m is the number pixels. We further assume that F is an information-reducing process of the following form

$$F = Q \circ T : \mathcal{P}^m \rightarrow \mathcal{R}^n, \tag{12.28}$$

where $\mathcal{R}$ is the integer dynamic range of the processed signal, $\mathbf{y}$ (i.e., $\mathbf{y} = F(\mathbf{x})$). The output signal, $\mathbf{y}$, is represented by an n-dimensional vector of integers, $\mathbf{y} \in \mathcal{P}^n$, where $m > n$. The transform, $T : \mathcal{P}^m \rightarrow R^n$, is a real-valued transformation that is subsequently followed by a quantization, Q. The operator, $Q : R^n \rightarrow \mathcal{R}^n$, is the rounding to the closest integer within the range, $\mathcal{R}$. The intermediate image, $T(\mathbf{x})$, is denoted by an n-dimensional vector, $\mathbf{u} \in R^n$.

The sender identifies *changeable* elements, $\mathbf{u}[j], j \in \mathcal{J}$, whose values are close to the middle of the rounding intervals, as illustrated in Figure 12.10. Mathematically, we have

$$\mathcal{J} = \{j | j \in \{1, \ldots, n\}, \mathbf{u}[j] \in [L + 0.5 - \epsilon, L + 0.5 + \epsilon]\} \text{ for some integer } L. \tag{12.29}$$

The parameter ϵ is called the tolerance and controls which cover elements will be selected as changeable. This threshold can be adaptive and depend on the neighborhood of each cover element, or even depend on a secret key if desired. The sender communicates a message to the receiver by rounding the changeable elements, $\mathbf{u}[j], j \in \mathcal{J}$, to either L or $L + 1$ and rounding all other elements, $\mathbf{u}[i], i \in \mathcal{J}$, in a regular manner (i.e., $\mathbf{y}[i] = Q(\mathbf{u}[i])$).

Examples of possible operations, F, that are suitable for perturbed quantization include downsampling, resampling, and, of course, JPEG compression. In terms of the mathematical operations defined above, for downsampling, the

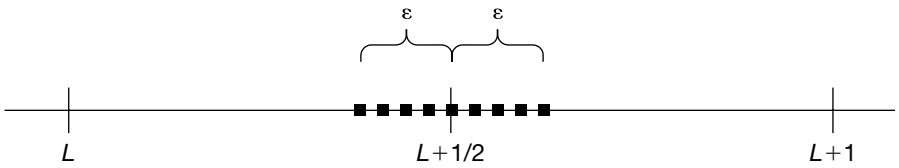


FIGURE 12.10

In perturbed quantization, values $\mathbf{u}[j]$ that fall in the dotted interval of length 2ϵ may be rounded to either L or $L + 1$.

transformation, T, maps a two-dimensional image array, $m_1 \times m_2$, of integers, $\mathbf{x}[i,j], i = 1, \ldots, m_1, j = 1, \ldots, m_2$, into an $n_1 \times n_2$ array of real numbers, $\mathbf{u} = \mathbf{u}[r,s], r = 1, \ldots, n_1, s = 1, \ldots, n_2, m_1 > n_1, m_2 > n_2$, using one of many resampling algorithms.

We now discuss the reasons why the nonshared selection rules used in perturbed quantization improve the security of steganographic schemes. To mount an attack, the warden must find statistical evidence that some of the values, $\mathbf{u}[j]$, have been quantized "incorrectly." However, for a number of reasons, this is difficult. First, the sender is using side information that is largely removed during quantization and is thus unavailable to the warden. Moreover, the rounding process at changeable elements is significantly influenced by the noise naturally present in images. This is less so for the remaining, unselected elements.

Nevertheless, the warden may be able to model some regions in the image sufficiently well (e.g., regions with a smooth gradient) that an attack on perturbed quantization becomes feasible. To avoid this, the sender can (and should) exploit additional selection rules to exclude from the changeable set, $\mathcal{J}$, those elements whose unquantized values can be predicted with high accuracy. There is no limit on the nature of these additional rules, since the sender and receiver can communicate using nonshared selection rules.

We note that the selection rule does not have to necessarily be of the type given in Equation 12.29, but can be defined differently based on heuristics depending on the format of the cover Work and properties of its elements. For example, in Fridrich *et al.* [148], the authors give an example of a slightly different selection rule for the situation when the information-reducing transformation is recompression of a cover JPEG image using a lower JPEG quality factor.

We now numerically evaluate the expected increase in distortion due to perturbed quantization steganography. Let us assume that the selection rule is of the form given by Equation 12.29. If the message bits are random, the result of embedding a message in the cover image, $\mathbf{x}$, is a probabilistic one-to-many mapping $[\mathbf{x} \rightarrow Q_\epsilon \circ T(\mathbf{x}) = \mathbf{y}']$, where $\mathbf{y}'$ is the stego image represented using an integer vector, $\mathbf{y}' \in \mathcal{R}^m$, and Q_ϵ is the perturbed quantizer, which is a probabilistic mapping

$$
\begin{aligned}
&Q_\epsilon(z) = L \quad \text{for } L \le z < L + 0.5 - \epsilon \\
&Q_\epsilon(z) = L + 1 \quad \text{for } L + 0.5 + \epsilon < z \le L + 1 \\
&Q_\epsilon(z) \in \{L, L+1\} \quad \text{with equal probability for } L + 0.5 - \epsilon \le z \le L + 0.5 + \epsilon.
\end{aligned}
$$

Note that $Q_\epsilon = Q$ for $\epsilon = 0$. The quantizers Q and Q_ϵ are identical with the exception of the interval $[L + 0.5 - \epsilon, L + 0.5 + \epsilon]$ where their outputs differ in 50% of the cases. It is straightforward to show that for a random variable, u, uniformly distributed on [0% , 1], the expected value of the quantization error, $|u - Q(u)|$, introduced by the regular rounding process, Q, is 1/4, while for the perturbed quantizer, Q_ϵ, it is $1/4 + \epsilon^2$. The average embedding distortion is the difference between the average error of both quantizers, which is ϵ^2. For

$\epsilon = 0.1$, this difference is at least one order of magnitude smaller than the average quantization error.

12.6 SUMMARY

This chapter has focused on a specific data-hiding application—steganography. The main points of this chapter are the following.

- As opposed to digital watermarking, the main property of steganography is statistical undetectability of embedded data. The payload is usually unrelated to the cover Work, which only serves as a decoy.
- The information-theoretic definition of steganographic security (Cachin's definition) is the most widely used definition in practice. It is usually applied in a simplified form by accepting a model for the cover Work.
- The simplest steganographic method—the LSB embedding—embeds messages by flipping the LSBs of individual elements of the cover Work. LSB embedding is not secure with respect to Cachin's definition because it introduces characteristic artifacts into the histogram of pixel values.
- Secure steganographic schemes must take into account steganalytic methods. One possibility is to replace the embedding operation of LSB flipping (F5) to avoid introducing easily detectable artifacts. Another possibility is to mask the embedding distortion as a naturally occurring phenomenon, such as to occur during image acquisition (stochastic modulation). Alternatively, we can design schemes that preserve some vital statistical characteristics of the cover image (OutGuess) or a *model* of the cover that is recoverable from the stego Work (model-based steganography).
- The embedding efficiency of steganographic schemes is defined as the average number of random message bits embedded using one embedding change. If the secret message is shorter than the embedding capacity, the embedding efficiency can be substantially improved using matrix embedding. This general embedding principle can be applied to the majority of steganographic schemes.
- In a typical steganographic scheme, the placement of embedding changes (the selection rule) is shared between the sender and the recipient. However, there are many situations when this information cannot be shared, such as in adaptive steganography, selection rules determined from side-information, or in public-key steganography. The problem of nonshared selection rules is equivalent to writing in memory with defective cells and can be efficiently approached using sparse linear codes, known as LT codes.

This page intentionally left blank

CHAPTER 13

Steganalysis

Steganalysis seeks to detect the presence of hidden messages in Works. In the previous chapter, we discussed steganographic security and introduced Cachin's information—theoretic definition of security. Intuitively, Cachin's result states that the more similar the two probability distributions of the cover and stego Works are, the more difficult it is to distinguish between cover and stego Works. We then described various approaches that have been developed that attempt, to one extent or another, to minimize the difference between the distributions of the cover and stego Works.

None of the existing practical steganographic algorithms achieves perfect security in the Cachin sense. Thus, in principle, all are detectable. In this chapter, we consider the various strategies of steganalysis that are available to the warden. In Section 13.1 we first discuss the various goals of the warden, the constraints under which the warden must operate, and the range of supplementary information the warden may have to assist in the task. Within this context, we then describe the various classes of steganalysis that the warden can use. Section 13.2 then describes a number of well-known steganalysis algorithms.

13.1 STEGANALYSIS SCENARIOS

The requirements for steganalysis can vary significantly depending on the operational scenarios. In this section, we discuss a number of issues and constraints that influence the choice of steganalysis algorithm.

Before proceeding, we remind the reader that there are three types of warden—passive, active, and malicious. The passive warden intercepts communications from Alice and tests for the presence of a hidden message. If no hidden message is detected, then the warden forwards the communications to Bob. Conversely, if a hidden message is detected, then the warden blocks the transmission to Bob (i.e., Bob receives neither the communication nor the hidden message).

An active warden is permitted more freedom. In particular, an active warden is free to modify the communication between Alice and Bob. Thus, even if the warden's steganalysis test fails, the communication may still be altered in an attempt to remove any hidden message that might evade detection. For example, if Alice and Bob are transmitting JPEG compressed images between one another, then the warden, Eve, might choose to recompress the images before forwarding them to Bob. In this way, any hidden message embedded with a JPEG-based steganographic algorithm is likely to be removed. Remember, that all of the steganographic algorithms described in the previous chapter assume a passive warden and are not designed to withstand modification to the cover Work (i.e., they are not robust).

A malicious warden may go one step further, in that Eve might attempt to impersonate Alice and send Bob false messages. To do so requires Eve to be capable of much more than the detection of a covert message. Eve must also know what steganographic algorithm Alice and Bob are using and any associated steganographic and encryption keys.

Thus, while the fundamental goal of the warden is to reliably detect the presence of a hidden message in communications between Alice and Bob, Eve may want to know much more. This higher level of analysis, which is more than just detection, is known as *forensic steganalysis* and is briefly discussed in Section 13.1.2. However, let us first consider the basic issue of detection.

13.1.1 Detection

The detection of a covert message hidden in a cover Work is most often modeled as a classification problem. A steganalysis algorithm receives Works as input and classifies them as either cover Works or stego Works. We will discuss this at length shortly. However, before doing so, it is worth considering what information is available to the warden.

At one extreme, Eve may know nothing and only have some level of suspicion that Alice and Bob are covertly communicating. At the other extreme, Eve may be certain that Alice and Bob are covertly communicating and even know the steganographic algorithm being used. Clearly, the former problem is more difficult than the latter. In the former case, Eve must develop a steganalysis algorithm that is capable of detecting all forms of steganography. Or, at the very least, a wide range of steganographic algorithms. This class of steganalysis algorithms is known as *blind steganalysis*. In the latter case, Eve must only be capable of detecting a specific steganographic algorithm; this class of detection algorithms is known as *targeted steganalysis*. In both cases, the problem is modeled as a classification problem and tools from pattern recognition, classification, and machine learning can be utilized. However, in some cases, Eve's knowledge of the steganographic algorithm provides an opportunity for a *system attack* much like the system attacks on watermarking schemes briefly mentioned in Chapter 10.

Both targeted and blind steganalysis model the problem of detection of a covert message as a classification problem. This is a particularly useful model, as we can exploit the very extensive results of research in this area. Broadly, classification divides the set of all possible objects into disjoint subsets where each subset forms a class. If there are only two classes, we often speak of "detection" rather than classification. The classifier is a mapping that depends on one or more parameters that are determined through training and based on the desired tradeoff between both types of error (false alarm and false detection, in the case of a two-class classifier) that the classifier can make. For example, if the objects being classified are represented by scalar values, as is frequently the case in targeted steganalysis, the classifier parameter is the threshold, which represents the boundary between the two classes. The location of this threshold will affect the two error rates and can be learned using a test set of images. In general, a classifier can have multiple parameters depending on its type and the dimensionality of the space in which objects are represented (i.e., the feature space). A useful introduction to classification techniques can be found in Duda *et al.* [116].

In theory, if we knew the probability distribution of cover and stego Works in the space of all possible Works, we could build an optimal classifier using the likelihood ratio test. The classifier can make two types of error—deciding that a cover Work is a stego Work, which is a false alarm, and deciding that a stego Work is a cover, which is a missed detection. The tradeoff between the probabilities of both errors is captured using Equations 12.4 and 12.5 from Section 12.3.1.

Unfortunately, in reality we cannot construct a classifier this way because the probability distributions of cover and stego Works will never be exactly known due to the high dimensionality of the space of all Works. To reduce the dimensionality, Works are represented using one or more *features* and the classifier is designed in this lower dimensional space.

Steganalysis thus begins with the choice of features with which to represent the Work. This is a key step as a judicious choice of features may permit very accurate classification. Alternatively, a poor choice of features may result in very poor classification.

The Fourier transform of the intensity histogram of a cover image is one simple example of a set of features. Here, we have a 256-dimensional feature vector, and each instance of the feature vector can be considered as a point in a 256-dimensional space.

In an ideal situation, we would like all feature vectors representing cover Works to map to a single point in the feature space, and for all stego Works to map to a different point in this same feature space. Classification is then trivial.

In practice, feature vectors representing cover Works will hopefully map to a region or cluster of the feature space and stego Works will map to another region or cluster of feature space. If these two regions do not overlap, then perfect classification is possible. More likely, there will be some degree of

overlap. The classification problem then becomes one of defining a boundary separating the two regions that minimizes the number of classification errors. One type of boundary is a hyperplane, but others are possible.

The more the two regions overlap, the more errors in classification occur. Minimizing this overlap is therefore critical, and is the goal of feature selection. To do so, we seek a representation that changes little across all the images in the cover distribution (i.e., the feature vectors of cover Works form a tight cluster). This same representation must be very different for stego Works, but again, should be stable across the distribution of stego Works (i.e., the feature vectors of stego Works are distant from those of cover Works) and also form a tight cluster.

One possibility to achieve these requirements is to estimate the cover Work from the stego Work. Of course, if we were able to accurately estimate the cover Work then detecting the embedding changes would be rather trivial—we could just compare the stego Work with the recovered cover Work. In practice, this is rarely possible. However, in Section 13.1.3, we give an example of a situation when a complete recovery of the cover Work is possible [143].

Usually, it is not possible to *exactly* recover the cover Work from the stego Work. Nevertheless, for some steganographic methods it is feasible to obtain an approximate estimate of the cover Work, which then can be utilized to derive useful features for steganalysis. The process of estimating the cover Work from the stego Work is called *calibration* (see Section 13.2.3). In Sections 13.2.3 and 13.2.4 we use calibration to construct features for a blind classifier.

In the next two sections, we explain the differences between the design of features for targeted and blind steganalysis, and in the last section, we deal with system attacks on steganographic schemes.

Targeted Steganalysis

In targeted steganalysis, we know the steganographic algorithm that is being used. This provides a powerful clue with which to choose the feature representation.

As an illustration of this, consider how the warden might proceed in building a classifier when she suspects that Alice and Bob are using least significant bit (LSB) embedding in the spatial domain to communicate secretly.

Flipping the LSBs of pixel values adds noise to the image and thus, on average, increases the differences between neighboring pixels. Thus, Eve might choose to calculate the sum of the absolute values of the differences between all pairs of neighboring pixels in the images, under the assumption that larger values of this sum would be indicative of LSB steganography. Unfortunately, since the variation of this measure across all naturally occurring images is very large, the clusters of cover and stego Work features will not be well separated. Textured or noisy images (images taken under low light conditions) will naturally have larger differences among neighboring pixels. Consequently, the warden will end up with an unreliable classifier exhibiting a high rate of false alarms.

A more clever warden may notice that the changes due to LSB embedding are not symmetrical. In fact, in LSB embedding, an even pixel value is never decreased (it is either increased or remains the same). Conversely, odd values are either decreased or remain the same. It is possible to exploit this asymmetry for the design of a targeted classifier. In Section 13.2.2, we show how this asymmetric property can be used to design an extremely reliable detector of LSB embedding. In fact, we will do much more than just construct a detector. We will also design an *estimator* of the number of embedding changes. Thus, the result of the steganalysis will not be just a binary decision as to the presence or absence of a hidden message, but a value that is strongly related to the length of the secret message.

In summary, the design of features for targeted steganalysis almost always begins with an analysis of the embedding algorithm to quantify the effect of embedding on the stego Work. Specific examples of targeted attacks are discussed in detail in Section 13.2.

Blind Steganalysis

Blind steganalysis has the same set of issues as targeted steganalysis, that is, the choice of feature representation and the determination of the decision boundary. However, the problem is made much more difficult by the fact that the steganographic algorithm being used by Alice and Bob is unknown.

Of course, presumably Eve does have a general knowledge of steganographic algorithms, and this knowledge can be used to help guide the choice of features. However, these features must not be just sensitive to a single steganographic algorithm, but to *all* steganographic algorithms.

Such a representation is unknown. And, if it were known, it would simply serve as a catalyst for the design of a new class of steganographic algorithm that did not significantly perturb the representation. Nevertheless, in practice, we must seek our best approximation to this ideal.

Given a feature representation, an example of which is described in Section 13.2.3, we must then train the classifier. Typically, during the training phase, a classifier needs two labeled sets, one of cover Works and the other of stego Works. However, for blind steganalysis we do not have the stego Works with which to train the classifier. Two options are open to us.

The first option is to train the blind steganalysis classifier using stego Works that are generated from a wide variety of known steganalysis algorithms. If we are lucky, one of these algorithms may be being used by Alice and Bob. If we are not so fortunate, we must hope that the training examples are sufficient for the classification algorithm to generalize to previously unseen steganographic algorithms.

The second option is to train the blind steganalysis classifier using only the cover Works. This is sometimes referred to as one-class learning. This classifier simply learns what a cover Work looks like in the feature space and then labels any Work that is incompatible with the distribution of covers as suspicious and

potentially a stego Work. This has the advantages of simplifying the training process, and the classifier may not need to be retrained when new embedding methods appear.

Blind classifiers that are designed to detect steganography in any embedding domain are called *universal* [24, 25, 132, 458]. Other blind classifiers are designed for a specific domain [166, 174], such as for steganographic methods that embed in the spatial domain or in quantized DCT coefficients of JPEG images [137, 330].

System Attacks

When high-definition DVD video players were introduced in 2006, they incorporated a much more advanced encryption algorithm [3] that, in principle, should have resisted attack. Nevertheless, within months, the system had been broken [7]. However, the attackers had not directly broken the encryption algorithm. Rather, they had discovered a weakness in the system implementation that allowed the secret key to be read from memory.

Steganographic algorithms can have similar weaknesses. For example, the electronic file representing the stego Work may contain specific markers or comments not typically found in such files. For example, the early release of the steganographic F5 algorithm always inserted the JPEG comment header, "JPEG Encoder Copyright 1998, James R. Weeks and BioElectroMech," which is rarely present in JPEG images produced by common image editing software. This comment can thus be used as a relatively reliable detector of F5-processed JPEG images. Johnson and Jajodia [210, 212] give examples of other unintentional fingerprints left in stego Works by specific stego products.

At this point, we would like to stress that the security of a steganographic algorithm depends not only on the embedding algorithm and the number of embedded message bits but also on the stego key. The stego key usually determines a pseudo-random walk through the image along which the message is embedded. A weak stego key creates a security weakness and can have grave consequences for the communicating parties. This is because the warden may run a dictionary attack for the stego key. For each key tried, Eve extracts an alleged message. And she will know that she has the correct key whenever she obtains a legible message. In this manner, the warden will have achieved the holy grail of steganalysis as she is now absolutely certain that Alice and Bob use steganography, and she will also know the message and the stego key. If the warden is malicious, she can then choose to impersonate either party, as well as simply block the communication.

If the message is encrypted before embedding, this brute-force search is not feasible because the warden has no means to distinguish between a random bitstream and an encrypted message. It may appear that as long as one uses a strong encryption scheme and encryption key, then the stego key does not have to be strong after all. However, this is not the case, because the warden can determine the stego key by means other than inspecting the message. In

particular, the warden can analyze the statistical properties of pixels along the pseudo-random walk determined by a stego key. For example, in simple LSB embedding, if the stego key is known, then the probability of encountering a modified pixel along the embedding path will be approximately 1/2 because half of the pixels are modified during embedding. However, when following a path generated from an incorrect key, a modified pixel is encountered only with a probability of $q/2$, where q is the relative message length. Thus, unless the Work is fully embedded, in which case $q = 1$, the statistical distribution of pixels will be different in the two cases. Eve can therefore use classical detection theory to construct a detector and search for the stego key [151].

13.1.2 **Forensic Steganalysis**

Assuming that the warden, Eve, can reliably determine when communications between Alice and Bob contain hidden messages, the next issue is what action does Eve want to take. Within the context of the Prisoners' Problem, as posed by Simmons [375], detection of a hidden message results in the warden blocking the communication between Alice and Bob, thereby preventing Bob from receiving the hidden message. However, this is a rather Draconian step, which may alert Alice and Bob to the fact that they are under surveillance. Moreover, in some scenarios, the warden may not have the authority or resources to block the communication channel. Instead, the warden might wish to read the message, or determine other related information. This is known as forensic steganalysis.

The warden can begin by trying to recover some attributes of the embedded message and properties of the stego algorithm. For example, the detection algorithm may provide Eve with an estimate of the number of embedding changes. In this case, she can approximately infer the length of the embedded message. If the approximate location of the embedding changes can be determined, this may point to a class of stego algorithms. For example, Eve can use the histogram attack, described in Section 13.2.1, to determine if the message has been sequentially embedded and thus narrow down the class of possible stego methods.

The character of the embedding changes also leaks information about the embedding mechanism. If Eve can determine that the LSBs of pixels were modified, she can then focus on methods that embed into LSBs. Eventually, Eve may guess which stego method has been used and attempt to determine the stego key and extract the embedded message. If the message is encrypted, Eve then needs to perform cryptanalysis on the extracted bitstream.

Based on the information available to the warden, Eve can mount different types of attacks. The most common case, which we have already considered, is the *stego Work only attack* in which Eve only has the stego Work. However, in some situations, Eve may have additional information available that may aid in her effort. For example, in a criminal case the suspect's computer may be

available with pairs of cover and stego Works on the hard disk. This will allow Eve to directly infer both the location and number of embedding modifications. This scenario is known as a *known cover attack*.

If the steganographic algorithm is known to Eve (e.g., the software is found on the suspect's computer), Eve can mount two more attacks—*the known stego method attack* and *the known message attack*. This facilitates a variety of opportunities for Eve. For example, to search for the stego key, she can now run a dictionary attack, embedding messages and comparing the location of embedding changes in the resulting stego Work to those in the stego Work that is under investigation. Depending on the stego method and the size of its key space, the dictionary attack may or may not be feasible.

13.1.3 The Influence of the Cover Work on Steganalysis

The choice of the cover Work has a major influence on the security of a steganographic system. For example, images with a low number of colors represented in palette image formats, such as GIF, provide little redundancy and thus steganographic embedding is usually more easily detectable. While this is intuitively obvious, there are some situations when it is much less apparent that a particular source of cover Works is inappropriate for steganography.

Consider the case when the cover Work is a JPEG image that has been decompressed back into the spatial domain, and a steganographic algorithm that embeds in the spatial domain (e.g., LSB embedding) is subsequently applied to the cover Work. The previous JPEG compression unintentionally creates a characteristic "signature" in the image, whose integrity is violated by the steganographic embedding. Specifically, during JPEG compression, an 8×8 block of pixels is mapped to an 8×8 block of quantized DCT coefficients. This mapping is obviously many-to-one. However, when embedding in the spatial domain of a decompressed JPEG image, it is very likely that the modified spatial block will not be compatible with any block of quantized DCT coefficients. In other words, no 8×8 block of quantized DCT coefficients can, when decompressed, produce the pixel values in the modified spatial block. Nevertheless, because the steganographic changes are small, the block will still retain strong traces of the previous JPEG compression. In fact, if the number of embedding changes is sufficiently small (on average less than one or two embedding changes per block), it is possible to recover the original block of cover Work pixels and identify which pixels have been modified! This is also an example of an unusual situation in which it is easier to detect a shorter message than a longer one.

Assuming that the designer of the stego system is aware of such serious pitfalls and avoids them, he or she still has a number of choices of how to select the source of cover Works. These choices affect the distribution of cover Works, which, in turn, affect the performance of all steganalysis algorithms [166, 227, 230]. Such choices include (1) the size of the images, (2) whether

the images are grayscale or color, (3) whether to use highly textured images and images with high-frequency noise, and (4) whether to use previously compressed imagery or images that have never been compressed. We briefly comment on some of these design choices.

In general, for a fixed relative message length, it is more difficult to detect a message in small images than in large images. This is because features computed from a shorter statistical sample are inherently more noisy. It is also generally easier to detect steganographic changes in color images than in grayscale images, because color images provide more data for statistical analysis and the steganalyst can utilize strong correlations between color channels. Detecting embedding changes in noisy or highly textured images is usually more difficult than in images with a low noise component and large smooth areas. This is because the cluster formed by features extracted from noisy or highly textured imagery is wider and thus creates an overlap with the cluster of stego Works. Scans of films or analog photographs are especially difficult for steganalysis because high-resolution scans of photographs resolve the individual grains in the photographic material, and this graininess manifests itself as high-frequency noise.

We now describe the image databases used to test the steganalysis methods in this chapter.

Database I consists of 2,567 raw, never-compressed, color images of different dimensions (all larger than 1 megapixel). Some are stored in 48-bit TIFF format, and others in 24-bit BMP format. They were acquired using 22 different digital cameras ranging from low-cost cameras to semi-professional cameras.

For all tests in this chapter, the images were converted to grayscale and, when needed, their color depth was reduced to 8 bits.

Database II has the same images as Database I, but compressed using a JPEG quality factor of 80.

Database III consists of 3,000 high-resolution never-compressed 1,500 × 2,100 scans of films in the 32-bit CMYK TIFF format. These were obtained from the NRCS Photo Gallery (http://photogallery.nrcs.usda.gov). For all tests, the images were converted to grayscale and downsampled to 640 × 480 using bicubic resampling.

13.2 SOME SIGNIFICANT STEGANALYSIS ALGORITHMS

In this section, we describe several well-known targeted and blind steganalysis algorithms. We begin with targeted methods. Sections 13.2.1 and 13.2.2 describe steganalysis of LSB embedding, since LSB embedding is the most common steganographic algorithm available over the Internet. Section 13.2.3 then describes calibration as a method for construction of features for blind steganalysis of JPEG images as well as for some targeted attacks. Section 13.2.4

on blind steganalysis in spatial domain also uses calibration to construct features but uses different principles.

13.2.1 LSB Embedding and the Histogram Attack

In Section 12.3.1, we showed that LSB embedding leaves characteristic artifacts in the histogram of pixel values. We can use this observation to construct a feature vector with which to mount a targeted steganalytic attack. The method is called the histogram attack [444] and is historically the first statistical attack described in the literature.

For LSB embedding, even pixel values are either left unmodified or increased by 1, while odd pixel values are either left unmodified or decreased. Thus, the grayscale values $(2i, 2i + 1)$ form a pair of values (PoV) that are exchanged into each other during embedding. This asymmetry in the embedding function can be exploited in the following manner.

Let $\mathbf{T}_c[j], j = 0, \ldots, 255$ denote the intensity histogram of the cover image and $\mathbf{T}_s$ be the corresponding histogram of the stego image after embedding qn bits, where n is the total number of pixels and $0 \leq q \leq 1$. Equivalently, we can say that we are embedding q bits per pixel (bpp) or that q is the *relative message length*.

Let us now calculate $\mathbf{T}_s$ as a function of $\mathbf{T}_c$ and q. Assuming that the secret message is a random bitstream, which is a reasonable assumption if the message is encrypted or compressed, the expected number of pixels with grayscale $2i$ that will be modified to $2i + 1$ is $\frac{q}{2}\mathbf{T}_c[2i]$. This is because on average one-half of the message bits will already match the LSB of the pixels. Similarly, the expected number of pixels with grayscale $2i + 1$ that will be changed to $2i$ is $\frac{q}{2}\mathbf{T}_c[2i + 1]$. Thus, we can write

$$E\{\mathbf{T}_s[2i]\} = \left(1 - \frac{q}{2}\right)\mathbf{T}_c[2i] + \frac{q}{2}\mathbf{T}_c[2i + 1] \tag{13.1}$$

$$E\{\mathbf{T}_s[2i + 1]\} = \frac{q}{2}\mathbf{T}_c[2i] + \left(1 - \frac{q}{2}\right)\mathbf{T}_c[2i + 1].$$

Note that, as already mentioned in Section 12.3.1, for a fully embedded image, $(q = 1)$, $E\{\mathbf{T}_s[2i]\} = E\{\mathbf{T}_s[2i + 1]\}$. In other words, the histogram bins $2i$ and $2i + 1$ will have approximately the same values, which will cause very obvious step artifacts in the histogram, as illustrated in Figure 12.2.

Note that the sum, $\mathbf{T}_s[2i] + \mathbf{T}_s[2i + 1] = \mathbf{T}_c[2i] + \mathbf{T}_c[2i + 1]$, is invariant under LSB embedding. The theoretically expected value for $\mathbf{T}_s[2i]$ is therefore $\overline{\mathbf{T}}_s[2i] = (\mathbf{T}_s[2i] + \mathbf{T}_s[2i + 1])/2$. Thus, it is possible to detect LSB embedding for $q = 1$ by testing whether $\mathbf{T}_s[2i] = \mathbf{T}_s[2i + 1]$.

Without any loss of generality, we can just use the even values of the grayscales and apply some statistical test that can verify that the even

values $\mathbf{T}_s[2i]$ follow the known distribution $\overline{\mathbf{T}}_s[2i]$. Here, we apply Pearson's chi-squared test [383] to determine whether $\mathbf{T}_s[2i] = \overline{\mathbf{T}}_s[2i]$. The chi-square test starts by calculating the chi-square test statistics S,

$$S = \sum_{i=1}^{k} \frac{(\mathbf{T}_s[2i] - \overline{\mathbf{T}}_s[2i])^2}{\overline{\mathbf{T}}_s[2i]}, \tag{13.2}$$

with $(k - 1) = 127$ degrees of freedom. Assuming that even grayscale values indeed follow the probability mass function, $\overline{\mathbf{T}}_s[2i]$, the test statistic, S, follows the chi-square distribution with $k - 1$ degress of freedom. We note that in practice, unpopulated bins (grayscales) must be merged so that $\overline{\mathbf{T}}_s[2i] > 4$ to ensure that S is approximately chi-square distributed.

Intuitively, a small value of S indicates that the data follows the expected distribution and we can conclude that the image contains a message embedded using LSB embedding. On the other hand, large values of the test statistic imply that no message is embedded. The statistical significance of S is measured using the so-called p-value, which is the probability that a chi-square distributed random variable with $k - 1$ degrees of freedom would attain a value larger than or equal to S:

$$p(S) = \frac{1}{2^{\frac{k-1}{2}}\Gamma\left(\frac{k-1}{2}\right)} \int_S^{\infty} e^{-\frac{x}{2}} x^{\frac{k-1}{2}-1} dx. \tag{13.3}$$

If the image does not contain a hidden message, S is large and $p(S) \approx 0$. In practice, we calculate a threshold value S_{tb} so that $p(S_{tb}) = \alpha$, where α is a chosen significance level. For a confidence level, α, we decide that a Work contains a secret message if $p(S) > \alpha$, or equivalently $S < S_{tb}$.

So far, the derivations have been carried out for a fully embedded image with $q = 1$ bpp. If the stego image was embedded sequentially but perhaps not fully, we can scan the stego image in the same order in which the message has been embedded and evaluate p for the set of visited pixels. After a short transient initial phase, the p-value will be close to 1 and it will suddenly fall to zero when we arrive at the end of the message and it will stay at zero until we exhaust all pixels (see Figure 13.2 for a sequentially embedded message with $q = 0.6$ for the cover image shown in Figure 13.1). This is because the test statistic S will cease to follow the chi-square distribution. Therefore, for sequential embedding this test not only determines, with a very high probability, that a random message has been embedded but it also estimates the length of the embedded message.

If the message-carrying pixels in the image are selected pseudo-randomly rather than sequentially, this test is ineffective unless the majority of pixels have been used for embedding. However, this analysis was generalized [339] by calculating Equation 13.2 over a sliding window of pixels in the image. The p-value will now fluctuate with a decreasing degree of fluctuation as the message length increases. This is because a randomly spread message will, due

FIGURE 13.1

Grayscale 2,269 × 1,701 cover image.

to chance, be more concentrated in some areas than in others. Another way to extend the histogram attack [443] to randomly spread messages is to inspect "hashes" of small groups of neighboring pixels or color channels of one pixel. The extended method is able to detect random LSB embedding for $q \geq 0.3$.

We now discuss a much more powerful attack on LSB embedding called sample pairs analysis.

13.2.2 Sample Pairs Analysis

The histogram attack on LSB steganography is based on the observation that in a fully embedded image, the number of pixels with grayscale $2i$ and $2i + 1$ should be approximately the same. This attack, however, cannot be applied when the image is only partially embedded along a pseudo-random path. Steganalysis methods that only use first-order statistics, such as histograms of pixels, ignore the dependency among neighboring pixels in natural images. More reliable and accurate detection algorithms are possible if the spatial correlation within images is utilized. *Sample pairs analysis* [121, 141] is an example of this.

Let $\mathcal{P}$ denote the set of all horizontally adjacent pixel pairs in the image. Let us partition $\mathcal{P}$ into three disjoint subsets, $\mathcal{X}, \mathcal{Y}$, and $\mathcal{Z}$, where

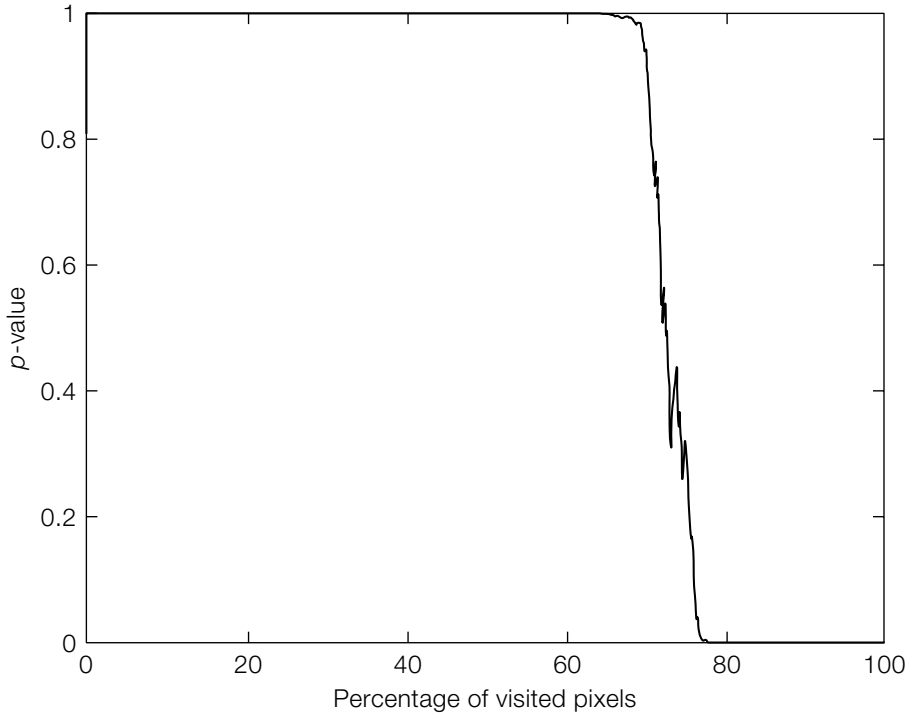


FIGURE 13.2

p-value as a function of the percentage of visited pixels.

$$\mathcal{X} = \{(u, v) \in \mathcal{P} | (v \text{ is even and } u < v) \text{ or } (v \text{ is odd and } u > v)\}$$
$$\mathcal{Y} = \{(u, v) \in \mathcal{P} | (v \text{ is even and } u > v) \text{ or } (v \text{ is odd and } u < v)\}$$
$$\mathcal{Z} = \{(u, v) \in \mathcal{P} | u = v\}.$$

Furthermore, let us partition the subset, $\mathcal{Y}$, into two subsets, $\mathcal{W}$ and $\mathcal{V}$, where $\mathcal{V} = \mathcal{Y} - \mathcal{W}$, and

$$\mathcal{W} = \{(u, v) \in \mathcal{P} | u = 2k, v = 2k + 1 \text{ or } u = 2k + 1, v = 2k\} \tag{13.4}$$

The sets $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{W}$, $\mathcal{V}$, and $\mathcal{Z}$ are called primary sets. Note that $\mathcal{P} = \mathcal{X} \cup \mathcal{W} \cup \mathcal{V} \cup \mathcal{Z}$.

It will soon become clear why this partitioning is useful for detecting LSB steganography. Let us analyze what happens to a given pixel pair (u, v) under LSB embedding. There are four possibilities:

1. Both u and v stay unmodified (modification pattern 00).

2. Only u is modified (modification pattern 10).

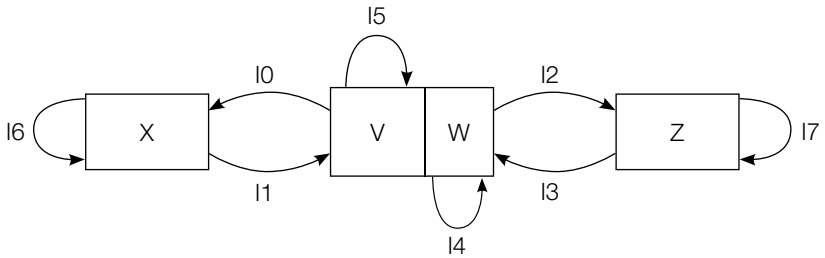


FIGURE 13.3

Transitions between primary sets under LSB flipping.

3. Only v is modified (modification pattern 01).

4. Both u and v are modified (modification pattern 11).

LSB embedding causes a given pixel pair to change its membership to one of the four primary sets. The arrows pointing from set $\mathcal{A}$ to set $\mathcal{B}$ in Figure 13.3 are labeled by the modification pattern, meaning that a pixel pair originally in $\mathcal{A}$ becomes a member of $\mathcal{B}$ if modified by the specified pattern during LSB embedding.

For each modification pattern $\pi \in \{00, 10, 01, 11\}$ and any subset $\mathcal{A} \subset \mathcal{P}$, let $\rho(\pi, \mathcal{A})$ be the fraction of pixel pairs in $\mathcal{A}$ modified with pattern π. If the message bits are randomly spread over the image, and are thus independent of the image content (nonadaptive embedding), then for each modification pattern $\pi \in \{00\,, 10\,, 01\,, 11\}$ and each primary set $\mathcal{A} \subset \mathcal{P}$, $\mathcal{A} \in \{\mathcal{X}\,, \mathcal{V}\,, \mathcal{W}\,, \mathcal{Z}\}$, we must have

$$\rho(\pi, \mathcal{A}) = \rho(\pi, \mathcal{P}). \tag{13.5}$$

If q is the relative message length, then the expected relative number of modified pixels is $q/2$. Using Equation 13.5, we thus have

$$\begin{aligned} \pi(00, \mathcal{P}) &= \left(1 - \frac{q}{2}\right)^2 \\ \pi(01, \mathcal{P}) &= \pi(10\,, \mathcal{P}) = \frac{q}{2}\left(1 - \frac{q}{2}\right) \\ \pi(11, \mathcal{P}) &= \left(\frac{q}{2}\right)^2. \end{aligned} \tag{13.6}$$

These transition probabilities and the set migration relationship from Figure 13.3 allow us to express the cardinalities of the primary sets after embedding as functions of q and the cardinalities before the embedding. Denoting the primary sets after embedding with a prime, we obtain

$$
\begin{aligned}
|\mathcal{X}'| &= |\mathcal{X}|\left(1-\frac{q}{2}\right)+|\mathcal{V}|q/2 \\
|\mathcal{Y}'| &= |\mathcal{Y}|\left(1-\frac{q}{2}\right)+|\mathcal{X}|q/2 \\
|\mathcal{W}'| &= |\mathcal{W}|\left(1-q+\frac{q^2}{2}\right)+|\mathcal{Z}|q\left(1-\frac{q}{2}\right).
\end{aligned}
\tag{13.7}
$$

Our goal now is to derive an equation for the unknown quantity, q, using only the cardinalities of primed sets, since they can be calculated directly from the stego image. However, to eliminate the unknown cardinalities of the primary sets of the cover image, we need an additional equation. We now note that on average

$$|\mathcal{X}| = |\mathcal{Y}|. \tag{13.8}$$

This assumption is true for natural images that typically contain a certain amount of noise, because it is equally likely to have $u > v$ or $u < v$ independent of whether u is even or odd.

The first two equalities in Equation 13.7 imply that

$$|\mathcal{X}'| - |\mathcal{Y}'| = (|\mathcal{X}| - |\mathcal{Y}|)(1-q). \tag{13.9}$$

Thus, since $|\mathcal{X}| = |\mathcal{Y}|$, we have $|\mathcal{X}| = |\mathcal{V}| + |\mathcal{W}|$, and from Equation 13.9, we have

$$|\mathcal{X}'| - |\mathcal{Y}'| = |\mathcal{W}|(1-q). \tag{13.10}$$

We see from Figure 13.3 that the embedding process does not modify the union $\mathcal{W} \cup \mathcal{Z}$. Denoting $\gamma = |\mathcal{W}| + |\mathcal{Z}| = |\mathcal{W}'| + |\mathcal{Z}'|$, and replacing $|\mathcal{Z}|$ with $\gamma - |\mathcal{W}|$, the expression of Equation 13.7 for $|\mathcal{W}'|$ becomes

$$|\mathcal{W}'| = |\mathcal{W}|(1-q)^2 + \gamma q\left(1-\frac{q}{2}\right). \tag{13.11}$$

Eliminating $|\mathcal{W}|$ from Equations 13.10 and 13.11 leads to

$$|\mathcal{W}'| = (|\mathcal{X}'| - |\mathcal{Y}'|)(1-q) + \gamma q\left(1-\frac{q}{2}\right). \tag{13.12}$$

Finally, since $|\mathcal{X}'| + |\mathcal{Y}'| + |\mathcal{Z}'| = |\mathcal{X}'| + |\mathcal{Y}'| + |\mathcal{W}'| + |\mathcal{Z}'| = |\mathcal{P}|$, Equation 13.12 is equivalent to

$$\frac{\gamma}{2}q^2 + (2|\mathcal{X}'| - |\mathcal{P}|)q + |\mathcal{Y}'| - |\mathcal{X}'| = 0. \tag{13.13}$$

All quantities in this equation can be calculated from the stego image (recall that $\gamma = |\mathcal{W}'| + |\mathcal{Z}'|$). The unknown message length, q, is obtained as the smaller root of this quadratic equation. If the equation has two complex conjugate roots, only their real parts should be taken. Also, if the smaller root is negative, one might output $p = 0$, as we know that p must be non-negative. Note that when $\gamma = 0$, it implies that $|\mathcal{X}| = |\mathcal{X}'| = |\mathcal{Y}| = |\mathcal{Y}'| = |\mathcal{P}|/2$, and the quadratic equation becomes an identity. In this case, we cannot calculate q. However, since γ is the number of pixel pairs that only differ in their LSBs, this will only happen very rarely for natural images.

INVESTIGATION

Sample Pairs Analysis

In this investigation, we test sample pairs analysis on images from Database I–III (Section 13.1.3).

System 21: SD_SPA

The SPA analysis accepts a grayscale image as input and returns an estimate of the embedded message length, q, using sample pairs analysis. The image is first scanned in a row-by-row order and then by columns. We register the total number of pairs (from both scans) for sets $\mathcal{X}', \mathcal{Y}', \mathcal{Z}', \mathcal{V}'$, and $\mathcal{W}'$ ($\gamma = |\mathcal{W}'| + |\mathcal{Z}'|$). After substituting the cardinalities of these sets into Equation 13.13, the estimated message length is obtained by solving the quadratic equation. If the roots are complex, only the real parts are taken. If $\gamma = 0$, an error message is issued that sample pairs analysis failed.

Experiment

Sample pairs analysis was tested separately on each database. Each cover image was embedded with random bitstreams of three different relative lengths, $q = 0.05$, 0.2, and 0.5. The average estimated message length and its standard deviation is shown in Table 13.1. Figures 13.4 and 13.5 show the histogram of the estimated message length for JPEG compressed images and scanned images. As expected, the estimator is most accurate for images that were JPEG compressed and for raw digital camera images. The low-pass character of JPEG compression obviously enables slightly more accurate estimation. For scanned imagery, the estimator exhibits a small positive bias and a significantly larger standard deviation compared to the other two image databases. This is because scans of films or photographs are inherently more noisy than images produced by digital cameras. Also, note that the distribution of estimated message length for all three databases is non-Gaussian, asymmetric, and exhibits long tails [229].

Table 13.1 Accuracy of estimated relative message length using sample pairs analysis for raw digital camera images, their JPEG compressed versions, and film scans, Database I, II, and III, respectively.

Database	Cover	0.05	0.25	0.5
I	0.0029 ± 0.0100	0.0528 ± 0.0096	0.2522 ± 0.0079	0.5015 ± 0.0063
II	0.0016 ± 0.0077	0.0515 ± 0.0075	0.2512 ± 0.0062	0.5009 ± 0.0053
III	0.0259 ± 0.0572	0.0746 ± 0.0544	0.2694 ± 0.0432	0.5129 ± 0.0291

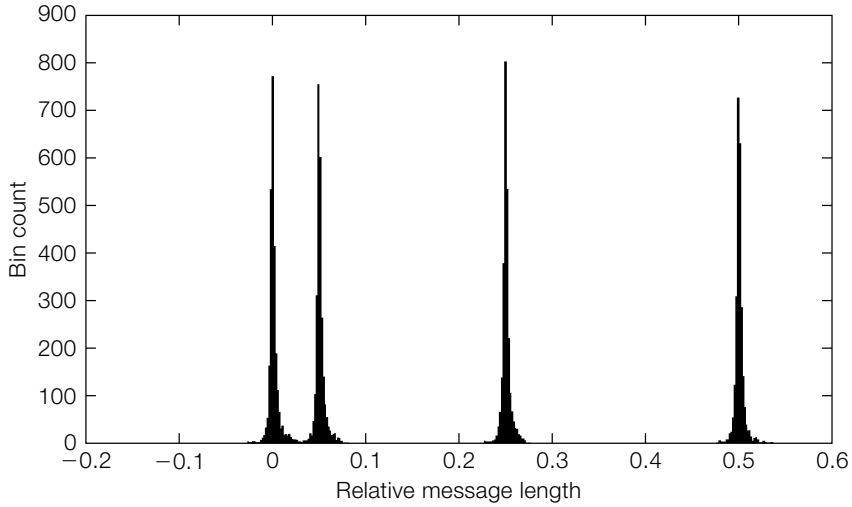


FIGURE 13.4

Histogram of the estimated message length for Database II compressed with a JPEG quality factor of 80%.

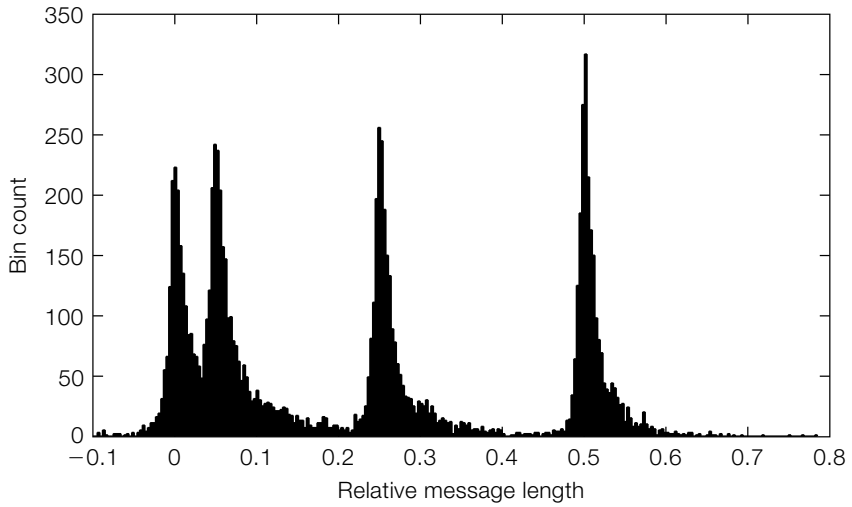


FIGURE 13.5

Histogram of the estimated message length for raw scans of films (Database III).

Sample pairs analysis was further improved in [122, 224, 226, 273] and extended to groups of more than two pixels in [120, 225], giving an even more accurate and reliable performance, especially for short messages. The statistical properties of the estimator were studied in [228, 229]. Other related approaches include [72, 104, 139, 146, 255, 465, 468, 469].

13.2.3 Blind Steganalysis of JPEG Images Using Calibration

The differences between embedding mechanisms in the JPEG and spatial domains suggest that much can be gained by designing separate blind steganalysis algorithms for each domain, and thereby exploiting specific properties of each class of steganographic algorithms. Here, we examine blind steganalysis in the JPEG domain, while Section 13.2.4 considers blind steganalysis in the spatial domain.

Virtually all steganographic methods designed for the JPEG format work by slightly manipulating the quantized DCT coefficients. Thus, features derived from the DCT domain likely will be more sensitive to steganographic embedding than features calculated from the spatial domain. Moreover, such features can provide a more straightforward interpretation of the influence of individual features on detection as well as valuable feedback to steganographers.

Blind steganalysis algorithms use a variety of image features, the size and sophistication of which are increasing with time. Features are often based on the first-order statistics of DCT coefficients (histograms) and second-order statistics (quantities depending on *pairs* of coefficients). However, many other features have been proposed. The interested reader is directed to [24, 132, 331, 370, 458] for further examples.

A common technique employed by blind steganalysis algorithms is the use of *calibration*, which attempts to estimate the original cover Work from the suspected stego Work. Of course, if this can be done accurately, then a larger difference is likely to indicate that the Work is a stego Work. And the difference between the estimated cover Work and the stego Work also enables us to estimate the number of embedding changes.

Estimating the cover Work from the stego Work is possible for JPEG images because the quantized DCT coefficients are robust to small distortions due to steganographic embedding. Calibration begins by decompressing the stego image to the spatial domain, then cropping the image by four columns, and then recompressing again using the same quantization matrix as that of the stego image. The resulting JPEG image is an estimate of the cover image that can be used to calibrate some macroscopic quantities, such as the DCT histogram, of the original image. Note that geometrical transformations other than cropping by four columns also can be used. For example, a slight rotation, resizing, or random warping as performed in Stirmark [247].

Why should recompression of a slightly geometrically distorted image give an estimate of the cover image? Heuristically, provided the quality factor of the

JPEG compression is not too low (e.g., lower than roughly 50), then the stego image is still very close to the cover image both visually and using distortion measures such as the PSNR. However, the spatial shift by 4 pixels and subsequent recompression effectively breaks the structure of the quantized DCT coefficients because the second compression "does not see" the first compression. As a result, certain macroscopic properties, such as the statistics of DCT coefficients, are similar to those of the original cover image.

To demonstrate how accurately calibration works, in Figure 13.6 we show the histogram of the DCT coefficient $(1, 2)$ of a cover image (+), stego image (∗) embedded with the F5 algorithm, and the estimate of the cover image histogram (∘) obtained using calibration. Clearly, calibration has provided a very close estimate to the original histogram. This estimate can be used for construction of features for blind steganalysis, as described below, or to mount a targeted attack on F5 and, in fact, on almost any steganographic algorithm that changes quantized DCT coefficients. This is because by analyzing the embedding mechanism, we can determine a relationship between the cover image histogram and the stego image histogram as a function of the embedded message length. The

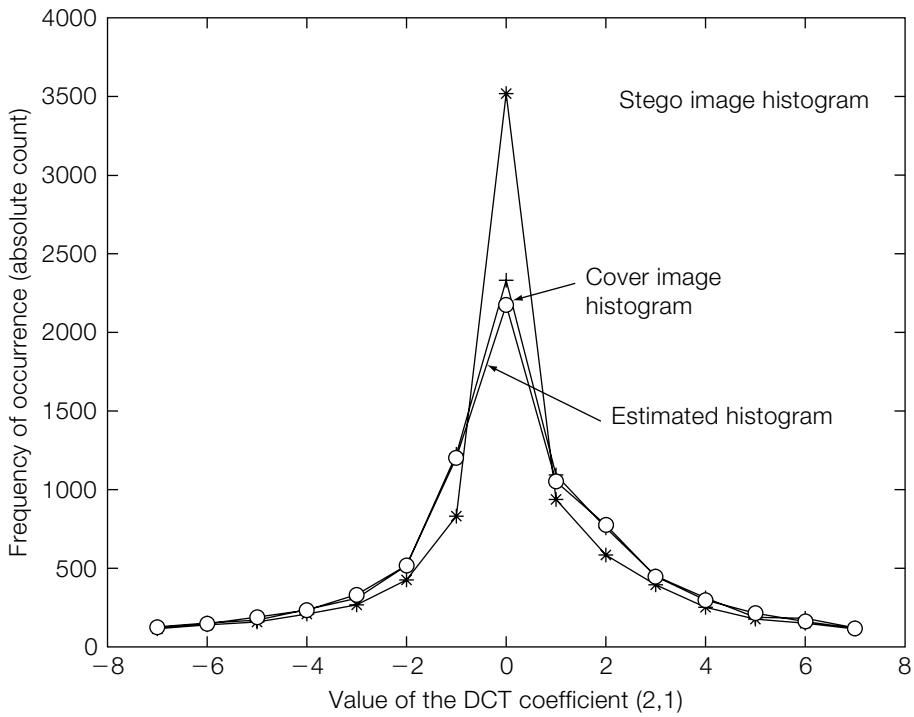


FIGURE 13.6

Histogram of the (1, 2) DCT coefficient of the cover image (+), F5 fully embedded stego image (∗), and the calibrated image (∘).

histogram of the stego image and the estimated cover image histogram then provide constraints from which the message length can be estimated [145] in the least square sense.

Besides the histogram of DCT coefficients, calibration can estimate many other macroscopic properties of the cover Work. Virtually all steganographic techniques for JPEG images increase the differences between pixels at the boundaries of 8×8 blocks. The sum of those differences is a higher-order statistic because it uses the fact that neighboring pixels in an image are correlated. We call this quantity "blockiness." The blockiness calculated from the stego Work and from the cover Work estimated by calibration form a useful feature for blind steganalysis. It also can be used for targeted attacks (e.g., the attack on OutGuess [145]).

In general, the process of calibration can be applied to the set of many other extracted features to decrease their variations across images and to increase their sensitivity to embedding. This is depicted in Figure 13.7. Construction of the calibrated features proceeds in the following manner. A selected functional, $\mathbf{F}$, is applied to the stego JPEG image, J_1. For example, this functional could be the histogram of all DCT coefficients. The stego image, J_1, is decompressed to the spatial domain, cropped by 4 pixels in each direction, and recompressed with the same quantization table as J_1 to obtain J_2. The same functional, $\mathbf{F}$, is then applied to the calibrated image, J_2. The calibrated feature vector, f, is then obtained as the difference (or an L_1 norm for vector or matrix functionals),

$$f = \|\mathbf{F}(J_1) - \mathbf{F}(J_2)\|_{L_1}. \tag{13.14}$$

Having specified the set of features, blind steganalysis then proceeds by selecting a classifier. There are a large number of classifiers available for this

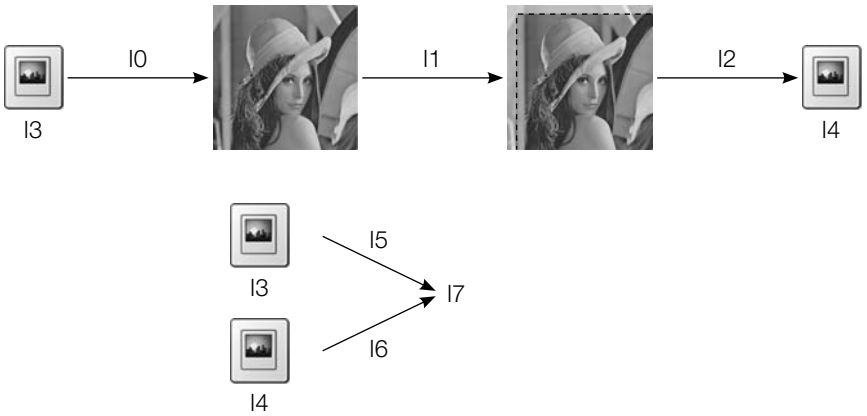


FIGURE 13.7

Estimation of cover image statistics via calibration. Calibrated features are obtained from (vector) functionals, **F**.

purpose. One of the simplest classifiers is the Fisher Linear Discriminant [117]. Nonlinear classifiers, such as support vector machines (SVMs) [424], typically provide the best performance and have become the method of choice for steganalysis [24, 132, 331].

After the classifier is chosen, it is presented with a training set of features extracted from cover and stego Works embedded with a specific stego algorithm. The purpose of training is to determine its internal parameters so that it can distinguish between both feature sets. The performance of the classifier is finally tested on a test set of cover and stego Works previously unseen by the classifier.

13.2.4 Blind Steganalysis in the Spatial Domain

Blind steganalysis in the spatial domain is similar to that in the JPEG domain. However, the features used are different and calibration based on the method described above is not possible.

Most steganographic methods in the spatial domain can be interpreted as adding noise with specific properties. Adding noise in the spatial domain corresponds to low-pass filtering of the image histogram, which can be readily seen as follows. Let $\mathbf{h}_c[i]$ and $\mathbf{h}_s[i]$ be the histogram of the grayscale cover and stego image, respectively. Assuming the stego signal added to the image is a random signal independent of the image with probability mass function $\mathbf{f}[j]$, $\sum_j \mathbf{f}[j] = 1$, the histogram of the stego image is a convolution

$$\mathbf{h}_s = \mathbf{h}_c * \mathbf{f}.$$

This is because if we consider the pixels of the cover image signal as instances of a random variable that is independent of the superimposed noise, the histogram (or empirical probability mass function) of the sum of the cover image signal and the noise signal is another random variable whose probability mass function is the convolution of probability mass functions of both signals.

Due to the low-pass character of the convolution, $\mathbf{h}_s$ will be smoother than $\mathbf{h}_c$ and thus will have more energy concentrated in low frequencies (see Figure 13.8). Thus, it is convenient to switch to the Fourier representation of the histograms and write for the Fourier-transformed quantities (denoted with corresponding capital letters)

$$\mathbf{H}_s[k] = \mathbf{H}_c[k]\mathbf{F}[k] \text{ for each } k. \tag{13.15}$$

The function $\mathbf{H}_s$ is called the histogram characteristic function (HCF) of the stego image, an example of which is illustrated in Figure 13.9.

The histogram characteristic function has been used in a variety of steganalysis algorithms [174, 227, 230]. Other features used for spatial domain steganalysis can be found in [24, 132, 443, 458].

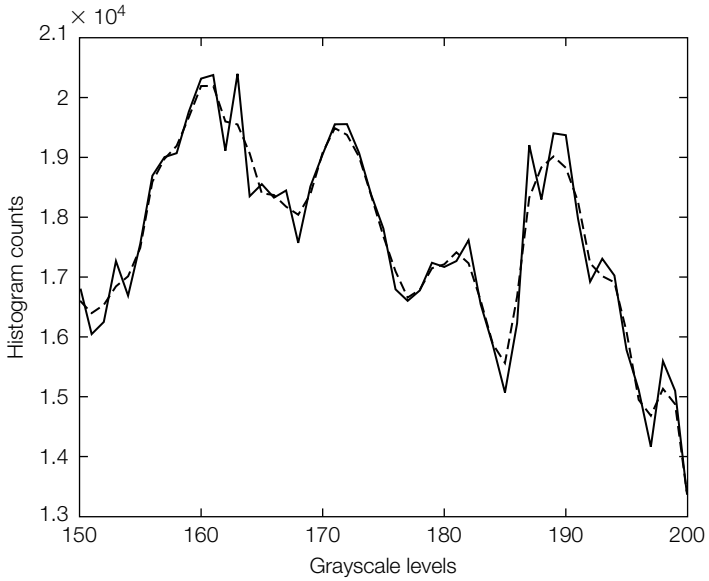


FIGURE 13.8

Histogram of cover and stego images after fully embedded with 1 embedding.

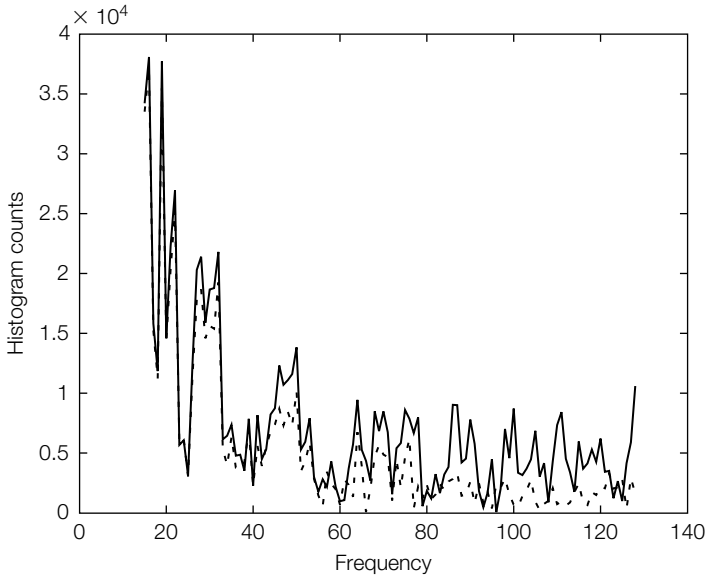


FIGURE 13.9

Histogram characteristic function of cover and stego images after fully embedded with ±1 embedding.

Ideally, we would like to also be able to estimate the cover Work and apply a similar calibration to features derived from the spatial domain. Since spatial domain steganography algorithms are often modeled as the addition of high-frequency noise, low-pass filtering or denoising algorithms [287] are commonly used for this purpose. The following investigation illustrates the use of denoising and highlights the sensitivity of steganalysis algorithms to variations in the statistics of images.

INVESTIGATION

Blind Steganalysis in the Spatial Domain Based on Denoising

An approach to blind steganalysis in the spatial domain is to calculate the features from the noise residual component of the image, r = s – *F*(s), obtained using a denoising filter, *F*. This automatically suppresses the influence of the cover image and thus enables construction of more sensitive features.

System 22: SD_DEN_FEATURES

We now describe a steganalytic method that uses a wavelet-based denoising filter described, in Mihcak *et al*. [287]. The image, s, under investigation is first transformed using an 8-tap Daubechis wavelet transform, *W*, such that $\mathbf{S} = W(\mathrm{s})$. Let $\mathbf{H}, \mathbf{D}, \mathbf{V}$ be the horizontal, diagonal, and vertical subbands in the lowest decomposition level. For example, if s is a 512×512 image, the three subbands have dimension 256×256. The denoising filter works by assuming that each subband can be modeled as an additive mixture of the image, which is modeled by a nonstationary Gaussian signal, $N(0, \sigma^2[i, j])$, and the noise, which is modeled as a stationary Gaussian signal, $N(0, \sigma_{\mathbf{n}}^2)$, with a known variance, $\sigma_{\mathbf{n}}^2$. The local variance $\sigma^2[i, j]$ is estimated from a square $w \times w$ neighborhood of (i, j),

$$\hat{\sigma}^2[i,j] = \max\left(0, \min\left(\sigma_3^2[i,j], \sigma_5^2[i,j], \sigma_7^2[i,j], \sigma_9^2[i,j]\right) - \sigma_{\mathbf{n}}^2\right),$$

where $\sigma_{\mathbf{w}}^2[i,j]$ is the local variance estimated from a w × w neighborhood of the wavelet coefficient $\mathbf{H}[i,j]$,

$$\sigma_w^2[i,j] = \frac{1}{w^2} \sum_{k,l=-\lfloor w/2 \rfloor}^{\lfloor w/2 \rfloor} (\mathbf{H}[i+k, j+l])^2.$$

The noise residual $\mathbf{r_H}$ is obtained by applying the Wiener filter,

$$\mathbf{r_H} = \mathbf{H}[i,j] - \frac{\hat{\sigma}^2[i,j]}{\hat{\sigma}^2[i,j] + \sigma_{\mathbf{n}}^2} \mathbf{H}[i,j].$$

The noise residuals, $\mathbf{r_D}$ and $\mathbf{r_V}$, can be obtained using similar formulas.

The steganographic features are calculated for each subband separately. The features used are the first 9 central absolute moments $\mu_k, k = 1,\ldots,9$ of the noise residual

$$\mu_k = \sum_{i,j} |\mathbf{r_H}[i,j] - \overline{\mathbf{r_H}}|^k.$$

Since there are three subbands, there will be total of 27 features for a grayscale image and 3×27 features for a color image.

The function DEN_FEATURES accepts a grayscale image represented as a matrix of pixels in the spatial domain and returns 27 features for construction of a blind steganalyzer. The features can be used to build a classifier using supervised training on a set of cover and stego images.

Experiment

To evaluate the algorithm, we first created a set of stego Works using ±1 embedding[1] in the spatial domain using different relative message lengths and three different image sets (Databases I, II, and III), described in Section 13.1.3. Classification was performed using the Fisher linear discriminant (FLD) [117] to show how separable the clusters of cover and stego images are.

The receiver operating characteristic (ROC) was obtained for each message length separately (i.e., for each message length, we trained a different FLD) to better demonstrate how the separability of clusters in the feature space decreases with decreasing message length. For a practical steganalyzer, we should train just one classifier on the set of cover images and the same number of stego images embedded with messages of various length.

The results, depicted in Figures 13.10, 13.11, and 13.12 are consistent with those obtained from sample pairs analysis in the Investigation in Section 13.2.2. Detection is significantly more reliable for digital camera images than for scans. This is due to the fact that scans exhibit significantly larger high-frequency noise. In particular, for high-resolution scans, the grains of the film are resolved, creating a high spatial frequency noise level. In contrast to the results from sample pairs analysis, the difference between uncompressed camera images and compressed images is now much more pronounced. The performance improvement on compressed imagery is due to lossy compression acting as a low-pass filter. Thus, in this case, the cover Works exhibit very little high-frequency noise, while the stego Works exhibit much more high-frequency noise due to the steganographic process of ±1 embedding.

[1] ±1 embedding is a simple modification of LSB embedding in which a pixel value is randomly modified by 1 or −1 (with probability 50%) to match its LSB with the message bit.

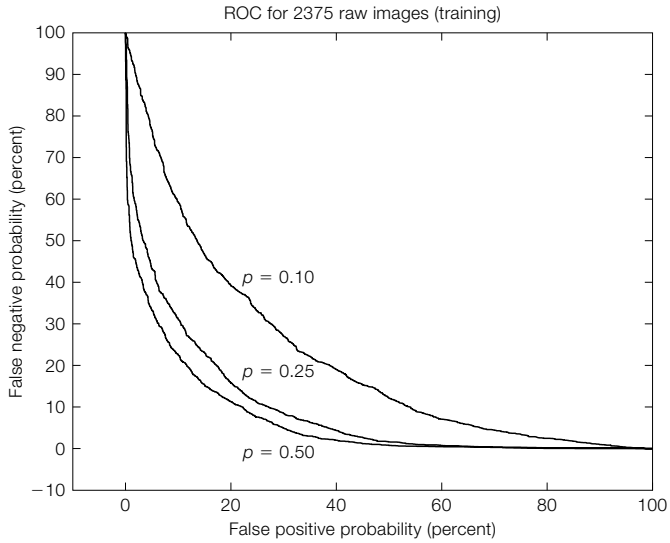


FIGURE 13.10

ROCs for detection of ±1 embedding in 2,567 raw digital camera images from Database I for relative message length $p = 0.1, 0.25, 0.50$.

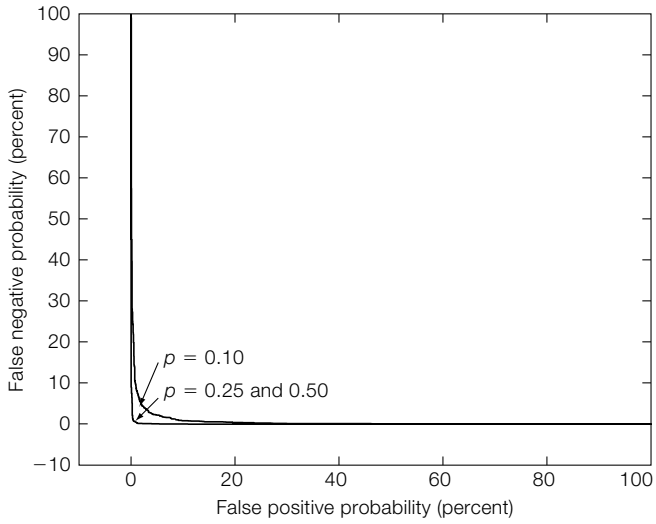


FIGURE 13.11

ROCs for detection of ±1 embedding in 2,567 digital camera images compressed using 80% quality JPEG (Database II) for relative message length $p = 0.1, 0.25, 0.50$.

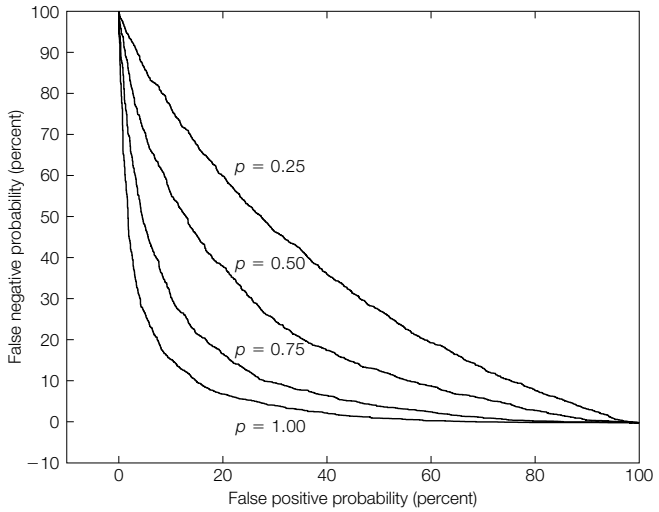


FIGURE 13.12

ROCs for detection of ±1 embedding in 3,000 raw scans of films (Database III) for relative message length $p = 0.25, 0.50, 0.75, 1.00$.

13.3 SUMMARY

This chapter has focused on steganalysis—ways to attack steganography. The main points of this chapter include the following.

- The goal of steganalysis is to detect the presence of a secret message.
- Forensic steganalysis aims at identifying the stego method and the stego key and recovering message attributes such as message length or content.
- There are two major classes of detection algorithms: targeted and blind.
- Targeted steganalysis is intended to detect a specific (target) steganographic algorithm.
- Blind steganalysis is intended to detect a wide range of steganographic algorithms, including previously unknown algorithms.
- Both targeted and blind steganalysis algorithms are classification problems and consequently results from pattern recognition and machine learning can be applied.
- Detection of steganalysis algorithms can also be accomplished at a system level by exploiting weaknesses in the implementation of specific algorithms.

- Sequential LSB embedding can be reliably detected using the histogram attack. Sample pairs analysis is an advanced steganalytic method targeted to LSB embedding along a pseudo-random path. Both methods can estimate the length of the embedded message.
- Calibration is a method for constructing detection statistics for both targeted and blind steganalysis. It is based on estimating the cover Work from the stego Work and is appropriate when the embedding is performed in the compressed (DCT) domain.
- For detection of steganographic algorithms that embed in the spatial domain, the histogram characteristic function (HCF) can be a useful feature.
- In the spatial domain, noise reduction filters form the equivalent of calibration.
- Performance of steganalysis algorithms can vary considerably depending on the source of the cover Works.

This page intentionally left blank

Glossary

2AFC See *Two Alternative, Forced Choice*.

1 embedding A steganographic system in which a *steganographic message* is embedded by modifying cover Work values by 1.

Absolute threshold (of hearing) The minimum audible intensity of a pure tone in a noiseless environment.

Active attack Any attempt to thwart the purpose of a watermarking system by modifying content. This includes *unauthorized removal* (or *removal attacks*) and *unauthorized embedding* (or *forgery attacks*).

Active warden A channel model for steganography in which the warden is intentionally modifying the transmitted messages.

Added pattern The sample-by-sample difference between an original and a watermarked version of a Work (i.e., the pattern added to a Work to embed a watermark).

Added vector/mark The difference between a marking-space vector extracted from a watermarked version of a Work and one extracted from the original version of that Work. This is analogous to an *added pattern,* which is the corresponding vector in media space.

Additive noise A random signal, independent of a Work, that is added to the Work.

Additive noise channel A communications channel in which the transmitted signal is corrupted by additive noise before reaching the receiver.

Adversary Anyone who attempts to thwart the purpose of a watermarking system. Depending on the application, adversaries might attempt a variety of attacks, including *unauthorized removal, unauthorized detection,* and *unauthorized embedding*. Other terms from the literature that have been used for an adversary include *pirate, hacker, attacker,* and *traitor*.

Alphabet A set of symbols, sequences of which can be used to represent messages.

Aspect ratio The ratio of width to height of an image or video frame. This ratio is sometimes presented as a fraction (e.g., 4/3), but more commonly as the width and height separated by a colon (e.g., 4:3).

Asymmetric fingerprinting A phrase sometimes used to refer to a special type of watermarking system designed for transaction tracking. In such a system, the owner of a Work transmits the Work to a buyer in such a way that the buyer receives a watermarked copy, but neither the owner nor the buyer knows what the watermark is. (This type of system is not discussed in this book.)

Asymmetric-key cryptography Any method of cryptography in which encryption and decryption require the use of different cipher keys.

Asymmetric-key watermarking Any method of watermarking in which embedding and detection require the use of different watermarking keys. The aim of such systems, as yet unattained, is to allow individuals to detect and decode a watermark without allowing them to embed or remove one.

Asymmetric watermarking See *Asymmetric-key watermarking*.

Attack Any attempt to thwart the purpose of a watermarking or steganography system.

Attacker See *Adversary*.

Authentication The process of verifying the integrity of a watermark or its cover Work. See also *Exact authentication* and *Selective authentication*.

Authentication mark A watermark designed to verify the integrity of the cover Work in which it is embedded.

AWGN Additive white Gaussian noise. This is an additive noise source in which each element of the random noise vector is drawn independently from a Gaussian distribution.

Base bits In an error correction code word, the bits from which the parity bits are computed. Generally, these bits are identical to the bit sequence the code word represents and are thus more commonly referred to as *message bits*. However, in a syndrome code, the message is encoded in the parity bits, and the base bits can take any value.

BCH Bose-Chandhuri-Hocquenghem. A form of error correction code.

BER See *Bit error rate*.

Bit error rate (BER) The frequency of bit errors when detecting a multibit watermark message.

Blind coding Any method of mapping messages into message marks that is independent of the cover Work in which the mark will be embedded. This is in contrast to *informed coding*.

Blind detection Detection of watermarks without any knowledge of the original, unwatermarked content. This is in contrast to *informed detection*. Some authors restrict *blind detection* to refer to detection without any key, as well as no knowledge of the original.

Blind embedding A simple approach to watermark embedding in which the *added pattern* is independent of the cover Work. This is in contrast to *informed embedding* and *informed coding*.

Blind steganalysis Steganalysis systems that are not targeted to a specific embedding mechanism.

Blind watermarking Watermarking systems that use *blind detection*. We avoid this term, preferring instead to specify whether we mean blind embedding or blind detection.

Block DCT A linear transform, commonly used in image processing. An image is first divided into blocks, and then the discrete cosine transform is applied to each block.

Brightness sensitivity The ability of the human eye to discriminate between different brightnesses of light. This generally changes as a function of mean brightness. Perceptual models account for this variable sensitivity when determining the visibility of a particular intensity change.

Broadcast monitoring An application of watermarking in which a detector monitors radio or television broadcasts, searching for watermarks. This can be used to ensure that advertisements are properly broadcast and that royalties are properly paid.

Cachin's definition of steganographic security Information-theoretic definition of *steganographic security*.

Channel capacity The maximum achievable code rate for a channel, or, more precisely, the supremium of all achievable rates. That is, any rate less than the channel capacity is achievable, but the channel capacity itself cannot be necessarily achieved.

Channel decoder A process that inverts the channel encoding process and attempts to identify and correct any transmission errors. This is performed at the receiver.

Channel encoder A function that maps messages into signals that can be reliably transmitted over a channel.

Chip In spread spectrum communications, a pseudo-random spreading sequence used to modulate a signal. In watermarking, chips may be considered analogous to reference patterns.

Cipher Any method of encrypting and decrypting messages.

Cipher key A secret key or pair of keys used for encryption and decryption of a message, as opposed to a *watermark key,* used for embedding and detecting messages in watermarks.

Ciphertext A message in encrypted form.

Cleartext A message that is not encrypted and can be understood by anyone.

Code book A set of *code words* or *code vectors*.

Code-division multiplexing The transmission of multiple messages or symbols over a single channel by representing them with orthogonal signals that might overlap in time, space, and/or frequency.

Code vector A vector that represents a message.

Code word A vector, sequence of symbols, or sequence of bits that represents a message.

Collusion attack A method of *unauthorized removal* in which an adversary obtains several versions of a Work, each with a different watermark. The Works are then combined to create a Work in which the watermarks are essentially undetectable. This term is sometimes also used to refer to attacks in which the adversary obtains many different Works that contain the same watermark, and uses them to identify the pattern associated with that mark.

Collusion-secure code A code that is secure against certain types of collusion attack. Specifically, it is secure against attacks that employ several copies of the same Work containing different watermark messages.

Compliant device A device that complies with a given standard for copy control technology. In the context of watermarking, a compliant device is a media recorder or player that detects watermarks and responds to them in appropriate ways.

Content The set of all possible Works of a particular type.

Contrast sensitivity The sensitivity of the human visual system to changes in luminance as a function of spatial frequency.

Convolution code A class of error correction codes. Equivalent to *trellis codes*.

Copy attack An attack in which an adversary copies a legitimate watermark from one Work to another. This constitutes a form of unauthorized embedding.

Copy control The prevention of copyright violations. This includes any system for preventing the recording of copyrighted content or playback of pirated content.

Copyright protection The legal protection afforded a Work by national and international law. Also used to refer to *copy control* technologies.

Correlation See *Linear correlation, Normalized correlation,* and *Correlation coefficient*.

Correlation coefficient A form of normalized correlation in which each vector is first modified to have zero mean prior to the magnitude normalization.

Coset A subset that results from partitioning a larger set.

Cover-<*datatype*> A cover Work. The *datatype* indicates the type of the Work. For example, "cover-text" or "cover-audio."

Covert communications Secret communication between two or more parties. Adversaries should not know that the communication is taking place.

Cover Work A Work in which a watermark is embedded (or is about to be embedded). The term is derived from the idea that the Work "covers" the watermark.

CPTWG Copy Protection Technical Working Group—an open committee founded in 1995 to propose a copy control system for DVD video.

Critical bandwidth A psychophysical property of hearing. Consider a narrowband noise source we perceive with a certain loudness. If the bandwidth of the noise source is increased, the perceived loudness will remain constant until the bandwidth exceeds the critical bandwidth. The critical bandwidth varies with frequency.

Cryptography The study and practice of keeping messages secure.

CSS Content scrambling system—a method of scrambling video used in the copy control system of DVD video.

Data hiding See *Information hiding*.

Data payload The number of bits a watermark encodes within a unit of time or within a Work.

DCT See *Discrete cosine transform*.

Decipher See *Decryption*.

Decoder See *Watermark decoder*.

Decryption The translation of ciphertext to plaintext.

DeCSS An illegal computer program that descrambles video protected with CSS.

Detection measure See *Detection statistic*.

Detection region A region of media space or marking space representing the entire collection of Works that yield positive watermark detections. Each watermark message has a distinct detection region.

Detection statistic Any measure of the likelihood that a signal is present. Common detection statistics include linear correlation, normalized correlation, and the correlation coefficient.

Detector See *Watermark detector*.

Device control The use of watermarks to control devices. This term encompasses more than just copy control applications. For example, watermarks have been used to synchronize a toy's actions with video.

DHSG Data Hiding Sub-Group—a subgroup of the *CPTWG* devoted to studying video watermarking technologies.

Digital rights management (DRM) Technical, legal, and business issues pertaining to copyright management and control when a Work is in a digital form.

Digital signature The digital equivalent of a traditional signature. They are used to verify the identity of the sender. A digital signature can be constructed by encrypting a one-way hash of a message with the sender's private key.

DIM See *Dithered index modulation*.

Direct message coding A method of coding watermark messages in which a separate reference mark is predefined for each message. The reference marks are usually generated pseudo-randomly. This is in contrast to *multisymbol message coding*.

Dirty-paper codes A class of codes in which each message is represented by a number of dissimilar code vectors. Named in reference to Costa's article, "Writing on Dirty Paper" [88]. These codes are instrumental in informed coding.

Discrete cosine transform A transform commonly used in image and video compression. The basic functions in this transform are real-valued cosine waves.

Distribution of unwatermarked Works Indicates the likelihood of each Work entering a watermark embedder or detector. Note that this distribution is application dependent.

Dithered index modulation A method of implementing a lattice code in which a Work is first correlated with a predefined set of patterns, and each correlation value is quantized to an integral multiple of some quantization step size.

Dithered quantization A process in which a *dither signal* is added to an original signal prior to quantization. The dither signal is usually pseudo-random. If a well-chosen dither signal is added to the input signal prior to quantization, the quantization error will be independent of the input signal.

Document See *Work*.

DRM See *Digital rights management*.

DWR Document-to-watermark ratio. This is a measure of the distortion introduced by the watermark embebber.

ECC See *Error correction code*.

Effectiveness The probability that a watermark embedder will successfully embed a watermark in a Work.

Elimination attack The removal of a watermark (by an adversary) so that no amount of subsequent postprocessing will retrieve it. This is in contrast to a *masking attack*.

Embedded-<*datatype*> A watermark. The *datatype* indicates what the watermark represents. For example, "embedded-text" or "embedded-image."

Embedder An algorithm or mechanism for inserting a watermark in a Work. Embedders take at least two inputs: the message to be embedded as a watermark and the cover Work in which we want to embed the mark. The output is a watermarked Work. Embedders may also employ a watermark key.

Embedding capacity Maximal number of bits that can be embedded in Work using a given steganographic system.

Embedding efficiency The number of embedded bits per unit distortion.

Embedding modification The process through which a cover Work is modified to embed a *steganographic message*.

Embedding region A region of media or marking space that represents the collection of all possible output Works from a watermark embedder.

Encipher See *Encryption*.

Encoder Abstractly: A mapping of messages into signals or code vectors. Concretely: An algorithm or mechanism that takes messages as input (usually represented with binary sequences) and outputs symbolic sequences or real-valued vectors. Encoders are typically

implemented in multiple stages. For example, the first stage—*error correction coding*—generates symbolic sequences that contain redundancy so that the messages can be identified even after errors are introduced. The second stage—*modulation*—maps symbolic sequences into real-valued vectors that can be physically transmitted (embedded in a watermark).

Encryption The translation of a plaintext message into ciphertext.

Erasable watermark A watermark that can be exactly removed from a Work, thereby obtaining a bit-for-bit copy of the original unwatermarked Work. Such watermarks are more commonly referred to as *invertible,* but this conflicts with another meaning of *invertible* in the context of proof of ownership (see *Invertible watermark*).

Error correction code (ECC) A mapping of messages into sequences of symbols such that not every possible sequence represents a message. In decoding such a code, sequences that do not correspond to messages are interpreted as corrupted code words. By defining the mapping between messages and code words in an appropriate way, it is possible to build decoders that can identify the code word closest to a given, corrupted sequence (i.e., decoders that correct errors).

Error correction encoder An encoder that implements an error correction code.

Exact authentication Verification that every bit of a given Work has remained unchanged. This is in contrast to *selective authentication*.

Extracted mark A vector obtained by applying an extraction function to a Work.

Extraction function A function that maps media-space vectors (content) into marking-space vectors.

Fading channel A type of communications channel in which the signal strength varies over time.

False negative A type of error in which a detector fails to detect a watermark in a watermarked Work.

False negative probability The probability that a *false negative* will occur during normal usage of a detector. Note that this is application dependent, in that "normal usage" is application dependent.

False positive A type of error in which a detector incorrectly determines that a watermark is present in a Work that was never watermarked.

False positive probability The probability that a *false positive* error will occur during normal usage of a detector. Note that depending on the application this might have subtly different meanings (see *Randomwatermark false positive probability* and *Random Work false positive probability*).

False positive rate The frequency with which *false positives* occur during normal usage of a detector.

Fidelity The perceptual similarity between an original Work and a Work that has undergone some processing, such as watermark embedding. This is in contrast to *quality*.

Fingerprinting In the context of watermarking, usually synonymous with transaction tracking. However "fingerprinting" sometimes refers to the practice of extracting inherent feature vectors that uniquely identify the content. We avoid using this term to prevent confusion.

Forgery See *Unauthorized embedding*.

Fragile watermark A watermark that becomes undetectable after even minor modifications of the Work in which it is embedded. These are unsatisfactory for most applications, but can be useful for authentication.

Frequency-division multiplexing The transmission of multiple messages or symbols over a single channel by representing them with patterns that are disjoint in the frequency domain.

Frequency-domain watermarking A watermarking system that modifies specific frequency coefficients of a Work. Note that the implementation of such a system does not necessarily require that the operations be performed in the frequency domain. Equivalent operations can be performed in the temporal or spatial domain.

Frequency hopping A method of spread spectrum communication in which the transmitter broadcasts a message by first transmitting a fraction of the message on one frequency, the next portion on another frequency, and so on.

Frequency sensitivity The response of a human perceptual system to differences in the frequencies of stimuli. In audition, the frequency of interest is the frequency of sounds. In vision, there are three distinct types of frequency sensitivity: sensitivity to spatial frequency, temporal frequency, and color.

Generational copy control The allowance of some copying while preventing subsequent copies from being made. This is of particular interest in the copy protection of content broadcast over the public airwaves, because under U.S. law consumers have the right to make a single copy of such content, but may not make a copy of the copy.

Golden ears People who possess extremely acute hearing.

Golden eyes People who possess extremely acute vision.

Gold sequences A specific set of binary sequences that have low cross-correlation with one another. See [358].

Hamming code An error correction code.

HAS Human auditory system.

Hash See *Hash function*.

Hash function A mapping of a variable-length string into a fixed-length string called a *hash*. Typically, the hash of a string is shorter than the original.

Histogram attack Attack on steganographic systems that exchange pairs of values such as *LSB embedding*, also known as "chi-square attack."

Histogram charateristic function Fourier transform of a histogram.

HVS Human visual system.

Illegitimate distortion A distortion that makes a Work invalid for some application. For example, in medical imaging, a distortion that might lead a doctor to an incorrect diagnosis. This is in contrast to *legitimate distortion*.

Imperceptible Undetectable by a human perceptual system. This is often defined statistically. See *Just noticeable difference*.

Implicit synchronization A class of methods for making watermarks robust against geometric transformations. In such methods, existing features of a Work are used for synchronization.

Information hiding The art and science of hiding information. The fields of steganography and watermarking are examples of information hiding, but the term covers many other subjects, such as anonymous communications and preventing unauthorized database inference.

Informed coding Any method of mapping messages into message marks that examines the cover Work during the coding process and determines the best code to use. In informed coding a given message can be mapped into different message marks for different cover Works. This is in contrast to *blind coding*.

Informed detection Detection of a watermark with some knowledge of the original unwatermarked Work. This knowledge can take the form of the original Work itself or some function of the original Work. Informed detection is in contrast to *blind detection*.

Informed embedding A form of embedding in which the modification of a message mark into an added mark is based on an examination of the cover Work. This includes perceptual shaping of a mark before embedding, as well as optimization of the added mark based on an estimate of robustness. It is in contrast to *blind embedding*.

Inseparability The property of watermarks in which they are not separated from their cover Works by conversion between formats or other forms of processing that preserve perceptual quality.

Invertible watermark A watermark that is susceptible to the ambiguity attack. This occurs if it is possible to generate a fake original from a distributed Work such that the distributed Work appears to be obtained by embedding a given watermark into the fake (i.e., it is possible to "invert" the embedding process). The term has also been used to describe what we refer to as an *erasable watermark*.

ITU International Telecommunications Union—a standards organization that, among other things, establishes guidelines for measuring television picture quality.

Jamming A term used in military communications to describe an adversary's effort to prevent a communications signal from being received.

JND See *Just noticeable difference*.

JPEG Joint Picture Experts Group—JPEG is a standard image compression technique based on block DCT quantization. JPEG 2000 is a multiscale wavelet-based image compression standard.

Just noticeable difference (JND) In psychophysics studies: A level of distortion that can be perceived in 50% of experimental trials. Multiple JNDs are defined a variety of ways.

Key See *Watermark key* and *Cipher key.*

Key management Procedures for ensuring the integrity of keys used in cryptographic systems. This can include key generation, key distribution, and key verification.

Keyspace The space from which keys are chosen. A large keyspace implies many keys and increased security because a larger space must be searched by an adversary.

Lattice code A form of dirty-paper code in which all code vectors lie on a lattice. *Dithered index modulation* and *least significant bit watermarking* employ lattice codes.

Least significant bit embedding The practice of embedding watermarks or secret messages by placing information in the least significant bits of their cover Works.

Legitimate distortion A distortion that does not change a Work's value for a given application. For example, in most applications lossless compression does not introduce perceptible artifacts. This is in contrast to *illegitimate distortion*.

Letterbox A format for displaying widescreen movies on standard television sets. In *letterbox* format, the image is reduced in size so that it fits onto the 4:3 television screen in its entirety, with some black lines added above and below.

Linear correlation A standard detection statistic. The linear correlation of two N-dimensional vectors, $\mathbf{u}$ and $\mathbf{v}$, is defined as

$$z_{\text{lc}}(\mathbf{u}, \mathbf{v}) = \frac{1}{N}\mathbf{u} \cdot \mathbf{v}.$$

Loudness sensitivity The ability of the human auditory system to distinguish between different loudnesses. This generally changes as a function of average loudness. Perceptual models account for this variable sensitivity when determining the audibility of a particular sound pressure change.

L_p-norm (Also called a *Minkowski summation*.) A class of measures of vector lengths. The L_p-norm of a vector, $\mathbf{x}$, is defined as

$$L_p(\mathbf{x}) = \sqrt[p]{\sum_i (\mathbf{x}[i])^p}.$$

The L_2-norm is a Euclidian length.

m-sequence A specific set of binary sequences that have low cross-correlation with one another. See [358].

MAC See *Message authentication code*.

Malicious warden Warden attacking a steganographic system by utilizing the specifics of the stego system, such as impersonating the communicating parties.

Marking space Any space in which watermark embedding and detection takes place. This may be a subspace or distortion of media space.

Masking A measure of an observer's response to one stimulus when a second stimulus is present.

Masking attack A form of watermark removal attack that makes watermarks undetectable using existing detectors, but does not prevent detection using more sophisticated detectors. In this sense, it does not truly remove the watermark. An example of a masking attack would be to imperceptibly rotate an image if existing detectors cannot correct for rotations. This is in contrast to *elimination attacks*.

Matrix embedding A coding scheme that increases *embedding efficiency*.

Media The means of representing, transmitting, and recording content. For example, an audio CD-ROM, a JPEG image file, or a VHS tape.

Media space A high-dimensional space in which each point corresponds to one Work.

Message Information to be transmitted over a channel or embedded in a watermark.

Message authentication code A one-way hash of a message that is then appended to the message. This is used to verify that the message is not altered between the time the hash is appended and the time it is tested.

Message bits See *Base bits*.

Message mark A vector in marking space that encodes a particular message.

Message pattern A watermark pattern that encodes a particular message. This is a vector in media space (i.e., a digital object of the same type and dimension as the cover Work).

Messaging layer A layer of a networking system concerned with composing and interpreting messages. This is in contrast to a *transport layer*.

Metric A function of two vectors, $d(\textit{texthbfx}, \mathbf{y})$, that satisfies the following conditions for all $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$:

- $d(\mathbf{x}, \mathbf{y}) \geq 0$
- $d(\mathbf{x}, \mathbf{y}) = 0$ if and only if $\mathbf{x} = \mathbf{y}$
- $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$
- $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y})$

Minkowski summation See *L_p-norm*.

Model-based steganography A steganographic system designed to preserve a given model of cover Work.

Modulator An algorithm or device that converts a sequence of symbols into a physical signal that can travel over a channel. For example, it might use its input to modulate the amplitude, frequency, or phase of a physical carrier signal for radio transmission. See also *Encoder*.

MPEG Motion Picture Experts Group—MPEG, MPEG-2, and MPEG-4 are standard video compression algorithms.

MP3 MPEG Layer 3 Audio—A standard audio compression algorithm.

Multimedia object See *Work*.

Multisymbol message coding A method of representing messages as sequences of symbols, which are then modulated. This is in contrast to *direct message coding*.

***N*-ball** A spherical volume in *N*-dimensional space. That is, where as *N*-sphere includes only the points on the surface, an *N*-ball includes the internal points as well.

***N*-cone** A conical region in *N*-dimensional space.

***N*-sphere** A spherical surface in *N*-dimensional space.

Nonshared selection rule A rule, which is not shared between the communicating parties, that is used to select individual elements of a cover Work to embed a *steganographic message*.

Nonsubtractive dither A method of reconstructing signals after *dithered quantization,* in which the dither signal is not subtracted before reconstruction. This is in contrast to *subtractive dither*.

Normalized correlation A standard detection statistic. The normalized correlation of two vectors, $\mathbf{u}$ and $\mathbf{v}$, is defined as

$$z_{\text{nc}}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{|\mathbf{u}||\mathbf{v}|}.$$

NTSC National Television Systems Committee—A television standard used in North America and Japan.

Oblivious In the context of watermarking, oblivious means blind. It is usually used to refer to blind detection.

One-way hash A hash function reasonably inexpensive to calculate, but prohibitively expensive to invert. That is, given an input string, it is easy to find the corresponding

output. However, given a desired output, it is virtually impossible to find a corresponding input string.

Panscan A format for displaying widescreen movies on standard television sets. In *panscan* format, the image fills the 4:3 television screen, but is cropped at the left and right.

Parity bits The bits of a code word that are added to the base bits during encoding.

Passive attack See *Unauthorized detection*.

Passive warden Warden who is just observing a channel without interfering with it in any manner.

Patchwork A watermarking algorithm described in Bander *et al.* [35].

Payload See *Data payload*.

Perceptually adaptive watermarking Watermarking systems that attempt to shape the added pattern according to some perceptual model.

Perceptually significant components Those components (features, frequencies, and so on) in which all but the most subtle changes will result in a loss of fidelity. Although at first it seems as though these are the components to avoid when watermarking, these are also the most generally robust components.

Perceptual shaping The practice of modifying a message mark according to the masking capabilities of a cover Work. This usually takes the form of attenuating the mark in areas where the cover Work cannot hide much noise, and amplifying it in areas where noise can be easily hidden.

Perceptual slack In a perceptual model: A measure of the relative amount by which a given component of a Work may be changed without serious perceptual impact.

Perturbed quantization steganography A steganographic system in which a secret message is embedded by perturbing a quantization process.

Pirate In general, an *adversary*. In the literature on transaction tracking, *pirate* is used more specifically to refer to a person who receives an unauthorized copy of a Work from a *traitor*.

Playback control Preventing playback of illegally recorded, copyrighted Works. A component of *copy control*.

Pooling In a model of perceptual distance: Combining the perceptibilities of separate distortions to give a single estimate for the overall change in the Work.

Power-spectrum condition A constraint on the spectrum of a watermark that has been derived in [392, 393] to provide robustness to Wiener filtering. The constraint states that the power spectrum of the added mark should closely match that of the cover Work.

Prisoners' problem A fictitious scenario motivating steganography in which prisoners need to secretly exchange messages without raising the attention of the warden.

Private watermarking The use of watermarking for any application in which a group called the *public* is not authorized to perform any watermarking operations, including watermark detection. There has been confusion in the literature between private watermarking and private-key cryptography, and between private watermarking and informed detection. For this reason, we avoid the use of this term.

Projected mark See *Extracted mark*.

Proof of ownership An application of watermarking in which a watermark is used to determine the owner of a cover Work.

Public-key cryptography See *Asymmetric-key cryptography*.

Public-key steganography A steganographic system in which the *stego key* is public and the *steganographic message* is encrypted using *assymmetric-key cryptography*.

Public watermarking This use of watermarking for any application in which a group called the *public* is authorized to detect the watermark, but is not authorized to perform any other watermarking operations. There has been confusion in the literature between public watermarking and public-key cryptography, and between public watermarking and blind detection. For this reason, we avoid the use of this term.

QIM See *Quantization index modulation*.

Quality An absolute measure of the goodness of a Work. This is in contrast to *fidelity*.

Quantization index modulation A method of watermarking in which each message is associated with a distinct vector quantizer. The embedder quantizes the cover Work (or a vector extracted from the cover Work) according to the quantizer associated with the desired message. The detector quantizes the Work using the union of all quantizers, and identifies the message. This is a method of implementing dirty-paper codes, described in Chen and Wornz [78]. It can be implemented with dithered index modulation.

Quantization watermarking Embedding watermarks by quantizing content in some domain. See, for example, *Dithered index modulation* and *Quantization index modulation*.

Random watermark false positive probability The probability that a randomly selected watermark is detected in a given Work.

Random Work false positive probability The probability that a given watermark is detected in a randomly selected Work.

Record control Preventing the recording of copyrighted content. A component of *copy control*.

Reference mark A vector in marking space used for comparison during watermark detection.

Reference pattern A pattern in media space used for comparison during watermark detection.

Reference vector See *Reference mark*.

Region of acceptable distortion See *Region of acceptable fidelity*.

Region of acceptable fidelity A set of points in media or marking space that represents all Works that appear essentially identical to a given cover Work. Each cover Work has a distinct region of acceptable fidelity.

Registration See *Synchronization*.

Removal attacks See *Unauthorized removal*.

Robustness The ability of watermarks to survive signal-processing operations. We draw a distinction between *robustness,* which refers to common operations, and *security,* which refers to hostile operations. Many authors do not draw this distinction and use robustness to refer to both types of operations.

Sample pairs analysis A method for estimation of the number of embedding changes introduced by *LSB embedding*.

SDMI Secure Digital Music Initiative—an organization specifying a copy-control system for digital music. The copy-control system itself is also referred to as SDMI.

Second-generation watermarking Strictly: Watermarking systems that employ some form of feature detection [245]. However, this term has been more loosely applied to any system that either does not embed watermarks by simply adding patterns or does not detect watermarks by simple linear or normalized correlation.

Secure transmission Transmission of messages in such a way that they are virtually guaranteed to arrive intact, even if an adversary is attempting to prevent it.

Security In watermarking, the ability of a watermark to resist intentional tampering. More generally, the ability of an entire system (which may incorporate watermarking) to resist intentional tampering.

Selection rule A rule used to select individual elements of a cover Work to embed a *steganographic message*.

Selective authentication Verification that a Work has not been subjected to a certain selected set of distortions. A selective authentication process detects application of *illegitimate distortions,* while ignoring application of *legitimate distortions*. This is in contrast to *exact authentication*.

Semi-fragile watermark A watermark that is fragile against certain distortions but robust against others. This is useful for *selective authentication*.

Sensitivity In perception: The human perceptual system's response to direct stimuli.

Sensitivity analysis attack An unauthorized removal attack possible when the adversary has a watermark detector but does not know the detection key. The watermark reference pattern is estimated by adding random patterns to a corrupted version of the Work and testing for detection. Once the reference pattern is estimated, it is subtracted from the Work to remove the watermark.

Side information Any information provided to either the transmitter or receiver in a communications system, other than the message to be transmitted or the received signal to be decoded. In watermarking, the cover Work is available to the embedder as side information.

Signal-to-noise ratio The power or amplitude of a given signal, divided by the power or amplitude of the noise that has been added to it. Usually expressed in decibels. In watermarking, either the watermark pattern or the cover Work can be considered the signal, the other being considered noise. This results in two different signal-to-noise ratios that might be reported: The *signal-to-watermark ratio* and the *watermark-to-noise ratio*.

Signal-to-watermark ratio The power or amplitude of a given cover Work divided by the power or amplitude of a watermark pattern added to it.

Signature See *Digital signature*.

Shaped pattern A message pattern that has been modified by perceptual shaping.

Slack See *Perceptual slack*.

SNR See *Signal-to-noise ratio*.

Sound pressure level The intensity of sound, in decibels (dB), relative to a reference intensity of 20 μ Pascals.

Space-division multiplexing The transmission of multiple messages or symbols over a single channel by representing them with patterns that are disjoint in the spatial domain.

SPL See *Sound pressure level*.

Spread spectrum The spreading of a narrowband signal over a much larger bandwidth.

Spread spectrum communications Communications in which narrowband signals are spread over much larger bandwidths. This was originally developed by the military to provide resistance to jamming and interference. The exact form of the spreading is a secret known only by the transmitter and receivers. Without knowledge of the spreading function, it is almost impossible for an adversary to detect or interfere with a transmission.

Steganalysis The art of detecting and decoding messages that have been hidden steganographically.

Steganographic capacity The maximal length of a *steganographic message* that can be *undetectably* embedded in a given Work.

Steganographic message A message the existence of which is kept secret by being hidden within a seemingly innocuous object.

Steganographic security Theoretical impossibility to detect existence of a *steganographic message*.

Steganography The art of concealed communication by hiding messages in seemingly innocuous objects. The very existence of a steganographic message is secret. This term is derived from the Greek words *steganos,* which means "covered," and *graphia,* which means "writing."

Steganology The art of steganography and steganalysis.

Stegoanalysis See *Steganalysis*.

Stego-<*datatype*> A Work containing a steganographic message. The datatype indicates the type of the Work. For example, "stego-text" or "stego-image."

Stego key A key required to detect a steganographic message.

StirMark A specific computer program that applies a variety of distortions to a watermarked Work to evaluate the robustness and security of the mark [328].

Stochastic modulation A steganographic system that embeds *steganographic messages* by adding noise of specified properties.

Structured codes Codes that can be efficiently represented and that allow efficient search for the closest code word.

Subtractive dither A method of reconstructing signals after *dithered quantization* in which the dither signal is subtracted before reconstruction. This is opposed to *non-subtractive dither*.

Symbol assignment function A rule that assignes a symbol from an *alphabet* to each element of the cover Work.

Symbols Unique elements that can be sequenced to represent messages.

Symmetric-key cryptography Method of encryption in which encryption and decryption require the use of the same secret cipher key.

Synchronization The process of aligning two signals in time or space.

Syndrome The pattern of differences between the parity bits in a received word and parity bits recomputed from the received base bits. This gives some indication of the errors in the received word. It is also used to encode messages in a *syndrome code*.

Syndrome code A class of structured code in which messages are represented in the syndromes of modified code words.

System attack An attack that exploits weaknesses in how watermarks are used, rather than weaknesses in the watermarks themselves.

Tamper resistance The ability of a watermark to resist tampering or attack. See *Security*.

Targeted steganalysis A steganalysis method attacking a specific steganographic system.

Telecine process The process of converting movies on film into electronic video.

Telltale watermarking The use of watermarks to try to identify processes that have been applied to a cover Work.

Time-division multiplexing The transmission of multiple messages or symbols over a single channel by representing them with patterns that are disjoint in the time domain.

Traitor In the literature on transaction tracking: A person who obtains a Work legally and subsequently misuses it. In this book, we do not differentiate between traitors and other types of adversaries.

Transaction tracking A watermarking application in which watermarks indicate transactions that have occurred in the history of a Work. In a typical transaction tracking application, the watermark identifies the first legal recipient of the Work. If it is subsequently discovered that the Work has been illegally redistributed, the watermark can help identify the person responsible.

Transparent watermark An imperceptible watermark. This is opposed to a *visible watermark*. We do not use these terms in this book because we do not discuss visible watermarks.

Transport layer A layer in a network model concerned with the transmission of messages, but not with their composition or interpretation. This is in contrast to a *messaging layer*.

Trellis code A class of error correction code in which messages specify paths through a graph known as a *trellis*. Each arc in the trellis is labeled with a sequence of symbols, and the sequences for arcs in a given path are concatenated to encode the message corresponding to that path.

Trellis-coded modulation A class of modulation techniques in which sequences of symbols specify paths through a trellis (see *Trellis code*). Each arc in the trellis is labeled with a vector, and the vectors for arcs in a given path are added to modulate the sequence corresponding to that path.

Trustworthy camera A camera that computes an authentication signature and appends it to the image.

Turbo code A class of error correction code.

Two Alternative, Forced Choice (2AFC) A classical experimental paradigm for measuring perceptual phenomena. For each trial stimulus, the observers are forced to choose one of two alternative responses.

Unauthorized deletion See *Unauthorized removal*.

Unauthorized detection A form of attack in which an adversary, who should not be permitted to detect and/or decode watermarks, nevertheless succeeds in detecting or decoding one.

Unauthorized embedding A form of attack in which an adversary, who should not be permitted to embed valid watermarks, nevertheless succeeds in embedding one.

Unauthorized reading See *Unauthorized detection*.

Unauthorized removal A form of attack in which an adversary, who should not be permitted to remove watermarks, nevertheless succeeds in removing one. This can take the form of either a *masking attack* or an *elimination attack*.

Unauthorized writing See *Unauthorized embedding*.

Undetectability Impossibility to detect the presence of embedded *steganographic message* in a Work.

Unobtrusive See *Imperceptible*.

Valumetric distortion A distortion that changes the values of individual samples in a Work.

VBI See *Vertical blanking interval*.

Vertical blanking interval That portion of a video signal that does not contain any picture content.

Visible watermark A visible mark placed over the content of an image or video. Usually, the mark has the properties that it is difficult to remove and is partially transparent. We do not discuss visible watermarks in this book.

Viterbi decoder A method of decoding trellis-coded or trellis-modulated messages.

Watermark A general term that can refer to either an embedded message, a reference pattern, a message pattern, or an added pattern. We use this term only informally.

Watermark decoder The portion of a watermark detector that maps extracted marks into messages. In most cases, this is the entire operation of the watermark detector. Therefore, in this book we generally make no distinction between a watermark detector and watermark decoder.

Watermark detector A hardware device or software application that detects and decodes a watermark.

Watermarking The practice of imperceptibly altering a Work to embed a message about that Work. However, note that some authors define this term in other ways. First, imperceptibility is not always considered a defining characteristic of watermarking. Second, the term "watermarking" is sometimes considered to cover only applications in which the same message is embedded in every copy of a Work (i.e., watermarking is not considered to include *transaction tracking*). In this case, the term "marking" is used to encompass both watermarking and transaction tracking.

Watermark key A secret key or key pair used for watermark embedding and detection. This key is analogous to the secret PN-sequences (chips) used in spread spectrum communications. A watermark key can be used in conjunction with a cipher key.

Watermark pattern In general: Either a reference pattern, a message pattern, or an added pattern, determined by context.

Watermark-to-noise ratio (WNR) The power or amplitude of a watermark pattern, divided by the power or amplitude of the cover Work to which it has been added. The denominator of this ratio may also include noise introduced by subsequent processing.

Watermark vector In general: Either a reference vector, a message mark, or an added vector, determined by context.

Wet-paper code A coding scheme that enables communication using a *nonshared selection rule*.

WNR See *Watermark-to-noise ratio*.

Work A song, video, picture, or any other object that can have watermarks embedded in it. This definition of the term *Work* is consistent with the language used in United States copyright law [416]. Other terms used to describe a Work in the literature include *document* and (*multimedia*) *object*.

References

[1] http://www.usatoday.com/tech/news/2001-02-05-binladen.htm.

[2] http://www.vernace.com.

[3] http://www.aacsla.com.

[4] http://www.tektronix.com/Measurement/App_Notes/25_14225/eng/25W_14225_2.pdf.

[5] Personal communication.

[6] http://www.research.att.com/njas/lattices/.

[7] http://forum.doom9.org/showthread.php?t=119871.

[8] Anonimity 0: Scientology 1, 1995. http://www.wired.com/wired/archive/3.05/eword.html?pg=7.

[9] In P. P. Ewald, editor. *50 Years of X-Ray Diffraction*. Reprinted in pdf format for the IUCr XVIII Congress, Glasgow, Scotland, Copyright 1962, 1999 International Union of Crystallography, 1999.

[10] Epson introduces revolutionary image authentication system for Epson digital cameras. *Business Wire*, April 5, 1999.

[11] J. Abbate. Inventing the Web. *Proc. of the IEEE*, 87(11):1999–2002, 1999.

[12] A. Abrardo, M. Barni. Informed watermarking by means of orthogonal and quasi-orthogonal dirty paper coding. *IEEE Trans. on signal Processing*, 53(2): 824–833, 2005.

[13] A. Abrardo, M. Barni, F. Perez-Gonzalez, and C. Mosquera. Trellis-coded rational dither modulation for digital watermarking. In *Proceedings of the 4th International Workshop on Digital Watermarking*, volume LNCS 3710, pages 351–360. Springer Lecture Notes in Computer Science, 2005.

[14] A. J. Ahumada and H. A. Peterson. Luminance-model-based DCT quantization for color image compression. *Proceedings of the SPIE*, 1666:365–374, 1992.

[15] A. M. Alattar. Reversible watermark using difference expansion of quads. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 377–380, May 2004.

[16] A. M. Alattar. Reversible watermark using the difference expansion of a generalized integer transform. *IEEE Transactions on Image Processing*, 13(8):1147–1156, August 2004.

[17] Masoud Alghoniemy and Ahmed H. Tewfik. Geometric distortion correction through image normalization. *Proc. of the International Conference on Multimedia and Expo*, 2000.

[18] E. Allamanche, J. Herre, O. Hellmuth, B. Bernhard Frobach, and M. Cremer. AudioID: Towards content-based identification of audio material. In *Proc. of the Int. Conf. on Web Delivering of Music*, 2001.

[19] O. Altun, G. Sharma, U. Celik, and F. Bocko. A set theoretic framework for watermarking and its application to semifragile tamper detection. *IEEE Trans. on Information Forensics and Security*, 1(4):479–492, 2006.

[20] R. Anderson, editor. *Information Hiding*, volume 1174 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[21] R. J. Anderson. Stretching the limits of steganography. In Ross Anderson, editor, *Information Hiding: First International Workshop*, volume 1174 of *Lecture Notes in Computer Science*, pages 39–48. Springer-Verlag, 1996.

[22] R. J. Anderson and F. A. P. Petitcolas. On the limits of steganography. *IEEE Journal of Selected Areas in Communications, Special Issue on Copyright and Privacy Protection*, 16(4):474–481, 1998.

[23] Daniel Augot, Jean-Marc Boucqueau, Jean-Francois Delaigle, Caroline Fontaine, and Eddie Goray. Secure delivery of images over open networks. *Proceedings of the IEEE*, 87(7):1251–1266, 1999.

[24] I. Avcibas, M. Kharrazi, N. Memon, and B. Sankur. Image steganalysis with binary similarity measures. *EURASIP Journal on Applied Signal Processing*, 17:2749–2757, 2005.

[25] I. Avcibas, N. D. Memon, and B. Sankur. Steganalysis using image quality metrics. In E. Delp, et al., editors, *Proceedings SPIE, Electronic Imaging, Security, and Watermarking of Multimedia Contents III*, San Jose, CA, 2001, volume 4314, pages 523–531, 2001.

[26] M. Backes and C. Cachin. Public-key steganography with active attacks. In J. Kilian, editor, *2nd Theory of Cryptography Conference TCC*, volume 3378 of *LNCS*, pages 210–226. Springer-Verlag, Heidelberg, 2005.

[27] F. Bacon. *Of the advancement and proficiencies of learning or the partitions of sciences*, volume VI. Leon Lichfield, Oxford, for R. Young and E. Forest, 1640.

[28] Z. Bahrav and D. Shaked. Watermarking of dithered halftoned images. In *Proc. of the SPIE Conf. on Security and Watermarking of Multimedia Data*, volume 3657, pages 307–316, 1999.

[29] M. Barni, F. Bartolini, V. Cappellini, and A. Piva. A DCT-domain system for robust image watermarking. *Signal Processing*, 66(3):357–372, 1998.

[30] James M. Barton. Method and apparatus for embedding authentication information within digital data. *United States Patent*, 5,646,997, 1997.

[31] P. Bas, J.-M. Chassery, and F. Davoine. A geometrical and frequential watermarking scheme using similarities. In *SPIE Conf. on Security and Watermarking of Multimedia Contents*, volume 3657, pages 264–272, 1999.

[32] P. Bas, J.-M. Chassery, and B. Macq. Robust watermarking based on the warping of predefined triangular patterns. *Security and Watermarking of Multimedia Contents II*, SPIE-3971:99–109, 2000.

[33] J. G. Beerends. Audion quality determination based on perceptual measurement techniques. In M. Kahrs and K. Brandenburg, editors, *Applications of Digital Signal Processing to Audio and Acoustics*, pages 1–38. Kluwer Academic Press, 1998.

[34] Alan E. Bell. The dynamic digital disk. *IEEE Spectrum*, 36(10):28–35, 1999.

[35] W. Bender, D. Gruhl, N. Morimoto, and A. Lu. Techniques for data hiding. *IBM Systems Journal*, 35(3/4):313–336, 1996.

[36] O. Benedens. Geometry-based watermarking of 3D models. *IEEE Computer Graphics and Applications*, 19:46–55, 1999.

[37] C. Berrou and A. Glavieux. Near optimum error correcting and decoding: turbo-codes. *IEEE Trans. on Communications*, 44(10):1261–1271, 1996.

[38] S. Bhattacharjee and M. Kutter. Compression tolerant image authentication. In *IEEE Int. Conf. on Image Processing*, volume 1, pages 435–439, 1998.

[39] J. Bierbrauer. On Crandall's problem. Personal communication, 1998.

[40] J. Bierbrauer. *Introduction to Coding Theory*. Chapman & Hall/CRC, 2004.

[41] K. A. Birney and T. R. Fischer. On modeling of DCT and subband image data for compression. *IEEE Trans. on Image Processing*, 4(2):186–193, 1995.

[42] J. A. Bloom, I. J. Cox, T. Kalker, J-P Linnartz, M. L. Miller, and B. Traw. Copy protection for DVD video. *Proc. of IEEE*, 87(7):1267–1276, 1999.

[43] J. Boeuf and J. P. Stern. An analysis of one of the SDMI candidates. In *Proceedings of Info Hiding '01*, 2001.

[44] B. P. Bogert, M. J. R. Healy, and J. W. Tukey. The *frequency analysis* of time series for echos: Cepstrum, pseudo-autocovariance, cross-cepstrum, and *saphe* cracking. *Proc. of the Symp. Time Series Analysis*, pages 209–243, 1963.

[45] D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data. In *Proceedings of Advances in Cryptology—CRYPTO'95*, volume Lecture Notes in Computer Science 963, pages 452–465, 1995.

[46] L. Boney, A. H. Tewfik, and K. N. Hamdy. Digital watermarks for audio signals. In *Proc. Third IEEE Int. Conf. on Multimedia Computing and Systems*, pages 473–480, 1996.

[47] A. Borodin, R. Ostrovsky, and Y. Rabani. Lower bounds for high dimensional nearest neighbor search and related problems. In *Proc. 31st ACM STOC*, pages 312–321, 1999.

[48] R. N. Bracewell. *The Fourier Transform and Its Applications*. McGraw-Hill, 1986.

[49] J. Brassil, S. Low, N. Maxemchuk, and L. O'Gorman. Electronic marking and identification techniques to discourage document copying. In *Proc. IEEE Infocom'94*, volume 3, pages 1278–1287, 1994.

[50] J. Brassil, S. Low, N. Maxemchuk, and L. O'Gorman. Electronic marking and identification techniques to discourage document copying. *IEEE Journal of Selected Areas in Communication*, 13:1495–1504, 1995.

[51] J. Brassil, S. Low, N. F. Maxemchuk, and L. O'Gorman. Electronic marking and identification techniques to discourage document copying. In *Proceedings, Infocom'*94, pages 1278–1287, 1994.

[52] G. W. Braudaway, K. A. Magerlein, and F. Mintzer. Protecting publicly available images with a visible image watermark. In *SPIE Conf. on Optical Security and Conterfeit Deterrence Techniques*, volume 2659, pages 126–133, 1996.

[53] Hans-Joachim Braun. Advanced weaponry of the stars. *American Heritage of Invention & Technology*, 12(4):10–16, 1997.

[54] D. Brewster. *Microscope*, volume XIV. Encyclopedia Britannica or the Dictionary of Arts, Sciences, and General Literature, Edinburgh, IX-Application of photography to the microscope, 8th edition, 1857.

[55] R. S. Broughton and W. C. Laumeister. Interactive video method and apparatus. *United States Patent*, 4,807,031, 1989.

[56] Janelle Brown. Playmate meets geeks who made her a net star. Wired News Web Site, May 1997. http://www.wired.com/news/culture/0,1284,4000,00.html.

[57] Lisa Gottesfeld Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, 1992.

[58] Redmond A. Burke. The most beautiful book of the fifteenth century: Hypnerotomachia poliphili. *Bulletin of the New York Public Library*, 58(9):419–428, 1954.

[59] Pierre Buser and Michel Imbert. *Audition*. MIT Press, 1992. Translated by R. H. Kay.

[60] C. Cachin. An information-theoretic model for steganography. In D. Aucsmith, editor, *Information Hiding, 2nd International Workshop*, volume 1525 of *LNCS*, pages 306–318. Springer-Verlag, New York, 1998.

[61] F. W. Campbell and J. J. Kulikowski. Orientation selectivity of the human visual system. *J. Physiol.*, 187:437–445, 1966.

[62] F. W. Campbell, J. J. Kulikowski, and J. Levinson. The effect of orientation on the visual resolution of gratings. *J. Physiol.*, 187:427–436, 1966.

[63] Carol Carr and Patricia E. O'Neill. Adding INSPEC to your chemical search strategy—let's get physical. *Database*, 18(2):99–102, 1995.

[64] D. Casasent and D. Psaltis. Position, rotation, and scale invariant optical correlation. *Applied Optics*, 15(7):1795–1799, 1976.

[65] D. Casasent and D. Psaltis. New optical transforms for pattern recognition. *Proc. of the IEEE*, 65(1):77–84, 1977.

[66] F. Cayre, C. Fontaine, and T. Furon. Watermarking security: Theory and practice. *IEEE Trans. on Signal Processing*, 53(10):3976–3987, 2003.

[67] F. Cayre, C. Fontaine, and T. Furon. A theoretical study of watermarking security. *Proceedings International Symposium on Information Theory (ISIT)*, pages 1868–1872, 2005.

[68] M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber. Lossless generalized LSB data embedding. *IEEE Transactions on Image Processing*, 14(2):253–266, February 2005.

[69] A. Chakrabarti, B. Chazelle, B. Gum, and A. Lvov. A lower bound on the complexity of approximate nearest neighbor searching on the Hamming cube. In *Proc. 31st ACM STOC*, pages 305–311, 1999.

[70] R. Chandramouli, M. Kharrazi, and N. Memon. Image steganography and steganalysis: Concepts and practice. In T. Kalker, Y. M. Ro, and I. Cox, editors, *Digital Watermarking, 2nd International Workshop, IWDW 2003, Seoul, Korea, October 20–22, 2003*, volume 2939 of *LNCS*, pages 35–49. Springer-Verlag, New York, 2004.

[71] R. Chandramouli, G. Li, and N. Memon. Adaptive steganography. In *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IV, San Jose, CA, January 21–24, 2002*, pages 69–78, 2002.

[72] R. Chandramouli and N. D. Memon. Analysis of LSB based image steganography techniques. In *Proceedings ICIP 2001, Thessaloniki, Greece, October 7–10, 2001*, 2001. CD Version.

[73] Ee-Chien Chang and M. Orchard. Geometric properties of watermarking schemes. *IEEE Int. Conf. on Image Processing*, 3:714–717, 2000.

[74] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[75] B. Chen and C.-E. Sundberg. Digital audio broadcasting in the FM band by means of contiguous band insertion and precanceling techniques. *IEEE Trans. on Communications*, 48(10):1634–1637, 2000.

[76] B. Chen and G. Wornell. Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. *IEEE Transactions on Information Theory*, 47(4):1423–1443, 2001.

[77] B. Chen and G. W. Wornell. Digital watermarking and information embedding using dither modulation. In *IEEE Second Workshop on Multimedia Signal Processing*, pages 273–278, 1998.

[78] B. Chen and G. W. Wornell. An information-theoretic approach to the design of robust digital watermarking systems. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1999.

[79] B. Chen and G. W. Wornell. Preprocessed and postprocessed quantization index modulation methods for digital watermarking. In *Security and Watermarking of Multimedia Contents II*, volume SPIE-3971, pages 48–59, 2000.

[80] Q. Cheng and T. S. Huang. Blind digital watermarking for images and videos and performance analysis. In *IEEE Int. Conf. on Multimedia and Expo*, volume 1, pages 389–392, 2000.

[81] Y. Cheng. Music database retrieval based on spectral similarity. In *Int. Symp. on Music Information Retrieval*, 2001.

[82] Jim Chou, S. Sandeep Pradhan, and Kannan Ramchandran. On the duality between distributed source coding and data hiding. *Thirty-third Asilomar Conference on Signals, Systems, and Computers*, 2:1503–1507, 1999.

[83] D. Coltuc and J.-M. Chassery. Distortion-free robust watermarking: A case study. In E. Delp III, editor, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505, January 2007.

[84] J. H. Conway and N. J. A. Sloane. *Sphere Packings, Lattices, and Groups*. Springer-Verlag, 1988.

[85] D. Coppersmith, F. Mintzer, C. Tresser, C. W. Wu, and M. M. Yeung. Fragile imperceptible digital watermark with privacy control. In *Security and Watermarking of Multimedia Contents*, volume SPIE-3657, pages 79–84, 1999.

[86] Corel Stock Photo Library 3, Corel Corporation, Ontario, Canada.

[87] Tom N. Cornsweet. *Visual Perception*. Academic Press, 1970.

[88] M. Costa. Writing on dirty paper. *IEEE Trans. Inform. Theory*, 29:439–441, 1983.

[89] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991.

[90] I. J. Cox. Spread spectrum watermark for embedded signalling. *United States Patent*, 5,848,155, 1998.

[91] I. J. Cox. Watermarking of image data using mpeg/jpeg coefficients. *United States Patent*, 6,069,914, 2000.

[92] I. J. Cox, T. Kalker, G. Pakura, and M. Scheel. Information transmission and steganography. In *Int. Workshop on Digital Watermarking*, 2005.

[93] I. J. Cox and J-P. M.G. Linnartz. Public watermarks and resistance to tampering. *IEEE Int. Conference on Image Processing*, 3, 1997.

[94] I. J. Cox and M. L. Miller. Counteracting geometric distortions for DCT based watermarking. *United States Patent*, 6,108,434, 2000.

[95] I. J. Cox, M. L. Miller, and A. McKellips. Watermarking as communications with side information. *Proc. IEEE*, 87(7):1127–1141, 1999.

[96] I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon. Secure spread spectrum watermarking for multimedia. *IEEE Trans. on Image Processing*, 6(12):1673–1687, 1997.

[97] I. J. Cox and M. L. Miller. A review of watermarking and the importance of perceptual modeling. In *Proceedings of SPIE, Human Vision & Electronic Imaging II*, volume 3016, pages 92–99, 1997.

[98] I. J. Cox, G. Doerr, and T. Furon. Watermarking is not cryptography. *Int. Workshop on Digital Watermarking*, 2006.

[99] S. Craver. On public-key steganography in the presence of an active warden. Technical Report RC 20931, IBM, 1997.

[100] S. Craver. On public-key steganography in the presence of an active warden. In *Proceedings 2nd Information Hiding Workshop*, pages 355–368, 1998.

[101] S. Craver, N. Memon, B.-L. Yeo, and M. M. Yeung. Can invisible watermarks solve rightful ownerships? IBM Technical Report RC 20509, IBM Research, July 1996. IBM CyberJournal: http://www.research.ibm.

[102] S. Craver, N. Memon, B.-L. Yeo, and M. M. Yeung. Resolving rightful ownerships with invisible watermarking techniques: Limitations, attacks and implications. *IEEE Journal of Selected Areas in Communication*, 16(4):573–586, 1998.

[103] G. Csurka, F. Deguillaume, J. J. K. O'Ruanaidh, and T. Pun. A Bayesian approach to affine transformation resistant image and video watermarking. In *Proc. of the 3rd Int. Information Hiding Workshop*, pages 315–330, 1999.

[104] O. Dabeer, K. Sullivan, U. Madhow, S. Chandrasekaran, and B. S. Manjunath. Detection of hiding in the least significant bit. *IEEE Transactions on Signal Processing*, 52:3046–3058, 2004.

[105] Scott Daly. The Visible Difference Predictor: An algorithm for the assessment of image fidelity. In A. B. Watson, editor, *Digital Images and Human Vision*, chapter 14, pages 179–206. MIT Press, 1993.

[106] G. Danezsis, R. Dingledine, and N. Mathewson. Mixminion: Design of a type iii anonymous remailer protocol. In *IEEE Proc. of the 2003 Symposium on Security and Privacy*, pages 2–15, 2003.

[107] Steve Decker. Engineering considerations in commercial watermarking. *IEEE Communications Magazine*, 39(8):128–133, 2001.

[108] J. F. Delaigle, C. De Vleeschouwer, and B. Macq. Watermarking algorithm based on a human visual model. *Signal Processing*, 66(3):319–335, 1998.

[109] D. Delannay and B. Macq. Generalized 2D cyclic patterns for secret watermark generation. In *IEEE Int. Conf. on Image Processing*, volume 2, pages 77–79, 2000.

[110] G. Depovere, T. Kalker, J. Haitsma, M. Maes, L. de Strycker, P. Termont, J. Vandewege, A. Langell, C. Alm, P. Norman, B. O'Reilly, G. Howes, H. Vaanholt, R. Hintzen, P. Donnelly, and A. Hudson. The VIVA project: Digital watermarking for broadcast monitoring. *IEEE Int. Conf. on Image Processing*, 2:202–205, 1999.

[111] Geert Depovere, Ton Kalker, and Jean-Paul Linnartz. Improved watermark detection using filtering before correlation. In *IEEE Int. Conf. on Image Processing*, volume 1, pages 430–434, Chicago, October 1998.

[112] Y. Desmedt. Subliminal-free authentication and signature. In *Advances in Cryptology, Proc. of EUROCRYPT '88*, volume 330 of *Lecture Notes in Computer Science*, pages 22–33. Springer, 1988.

[113] Jana Dittmann, Thomas Fiebig, and Ralf Steinmetz. A new approach for transformation invariant image and video watermarking in the spatial domain: SSP self-spanning

patterns. *Security and Watermarking of Multimedia Contents II*, SPIE-3971:176–185, 2000.

[114] Ray Dolby. Apparatus and method for the identification of specially encoded FM stereophonic broadcasts. *United States Patent*, 4,281,217, 1981.

[115] M. Dolson. The phase vocoder: A tutorial. *Comput. Music J.*, 10:14–27, 1986.

[116] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification, second edition*. John Wiley & Sons, Inc., 2001.

[117] R. O. Duda, P. E. Hart, and D. H. Stork. *Pattern Classification*. Wiley Interscience, New York, 2000.

[118] J.-L. Dugelay and F. A. P. Petitcolas. Possible counter-attacks against random geometric distortions. In *Proc. SPIE Conf. on Security and Watermarking of Multimedia Content II*, volume 3971, pages 338–345, 2000.

[119] I. Dumer. Concatenated codes and their multilevel generalizations. In V. S. Pless, W. C. Huffman, and R. A. Brualdi, editors, *Handbook of Coding Theory: Volume II*, chapter 23, pages 1911–1988. Elsevier Science, Amsterdam, 1998.

[120] S. Dumitrescu and X. Wu. LSB steganalysis based on higher-order statistics. In *Proceedings ACM Multimedia and Security Workshop, New York, NY, August 1–2*, 2005.

[121] S. Dumitrescu, X. Wu, and N. Memon. On steganalysis of random LSB embedding in continuous-tone images. In *Proceedings ICIP, Rochester, NY, September 22–25, 2002*, pages 324–339, 2002.

[122] S. Dumitrescu, X. Wu, and Z. Wang. Detection of LSB steganography via sample pair analysis. In F. A. P. Petitcolas, editor, *Information Hiding, 5th International Workshop, IH 2002, Noordwijkerhout, The Netherlands, October 7–9, 2002*, volume 2578 of *LNCS*, pages 355–372. Springer-Verlag, New York, 2003.

[123] J. Eggers, R. Buml, and B. Girod. A communications approach to steganography. In *Proceedings SPIE, Electronic Imaging, Security, and Watermarking of Multimedia Contents IV, San Jose, CA, January 21–24, 2002*, volume 4675, pages 26–37, 2002.

[124] J. J. Eggers, J. K. Su, and B. Girod. Robustness of a blind image watermarking scheme. In *IEEE Int. Conf. on Image Processing*, volume 3, pages 17–20, 2000.

[125] Joachim J. Eggers, Robert Buml, Roman Tzschoppe, and Bernd Girod. Scalar costa scheme for information embedding. *IEEE Transactions on Signal Processing*, 51(4):1003–1019, April 2003.

[126] Joachim J. Eggers and Bernd Girod. Watermark detection after quantization attacks. In A. Pfitzmann, editor, *Third Int. Workshop on Information Hiding*, volume 1768 of *Lecture Notes in Computer Science*, pages 172–186, 1999.

[127] Joachim J. Eggers and Bernd Girod. Quantization effects on digital watermarks. *Signal Processing*, 81(3), 2001.

[128] Joachim J. Eggers, Jonathan K. Su, and Bernd Girod. Public key watermarking by eigenvectors of linear transforms. In *EUSIPCO*, Tampere, Finland, Sept. 2000.

[129] J. Eggers, R. Baumi, and B. Girod. Estimation of amplitude modifications before SCS water mark detection. *Proc. SPIE Security, Steganography and Watermarking of Mutlimedia Contents*, 387–398, 2002.

[130] F. Ergun, J. Kilian, and R. Kumar. A note on the limits of collusion-resistant watermarks. In *Advances in Cryptology—EUROCRYPT '99*, pages 140–149. Springer-Verlag, 1999.

[131] J. M. Ettinger. Steganalysis and game equilibria. In D. Aucsmith, editor, *Information Hiding, 2nd International Workshop, IH 1998, Portland, Oregon, April 1998*, volume 1525 of *LNCS*, pages 319–328. Springer-Verlag, New York, 1998.

[132] H. Farid and L. Siwei. Detecting hidden messages using higher-order statistics and support vector machines. In F. A. P. Petitcolas, editor, *Information Hiding, 5th International Workshop, IW 2002, Noordwijkerhout, The Netherlands, October 7–9, 2002*, volume 2578 of *LNCS*, pages 340–354. Springer-Verlag, New York, 2002.

[133] Aníbal J. S. Ferreira. An odd-DFT based approach to time-scale expansion of audio signals. *IEEE Trans. Speech and Audio Proc.*, 7(4):441–453, 1999.

[134] David J. Field. Relations between the statistics of natural images and the response properties of cortical cells. *J. Opt. Soc. Am. A*, 4(12):2379–2394, 1987.

[135] D. Fragoulis, G. Rousopoulos, T. Panagopoulos, C. Alexiou, and C. Papaodysseus. On the automated recognition of seriously distorted musical recordings. *IEEE Trans. on Signal Processing*, 49(4):898–908, 2001.

[136] E. Franz. Steganography preserving statistical properties. In F. A. P. Petitcolas, editor, *Information Hiding, 5th International Workshop*, volume 2578 of *LNCS*, pages 278–294. Springer-Verlag, New York, 2002.

[137] J. Fridrich. Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes. In J. Fridrich, editor, *Information Hiding, 6th International Workshop, IW 2004, Toronto, Canada, May 23–25, 2004*, volume 3200 of *LNCS*, pages 67–81. Springer-Verlag, New York, 2005.

[138] J. Fridrich and M. Goljan. Images with self-correcting capabilities. In *Proc. IEEE Int. Conf. on Image Processing*, volume 3, 1999.

[139] J. Fridrich and M. Goljan. On estimation of secret message length in LSB steganography in spatial domain. In *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VI, San Jose, CA, January 19–22, 2004*, volume 5306, pages 23–34, 2004.

[140] J. Fridrich, M. Goljan, and M. Du. Invertible authentication. In *Proc. of SPIE, Security and Watermarking of Multimedia Contents*, 2001.

[141] J. Fridrich, M. Goljan, and R. Du. Detecting LSB steganography in color and grayscale images. *IEEE Multimedia, Special Issue on Security*, 8(4):22–28, October–December 2001.

[142] J. Fridrich, M. Goljan, and R. Du. Reliabile detection of LSB steganography in grayscale and color images. In J. Dittmann, K. Nahrstedt, and P. Wohlmacher, editors, *Proceedings of the ACM, Special Session on Multimedia Security and Watermarking, Ottawa, Canada, October 5, 2001*, pages 27–30, 2001.

[143] J. Fridrich, M. Goljan, and R. Du. Steganalysis based on JPEG compatibility. In A. G. Tescher, editor, *Special Session on Theoretical and Practical Issues in Digital Watermarking and Data Hiding, SPIE Multimedia Systems and Applications IV, Denver, CO, August 20–24, 2001*, volume 4518, pages 275–280, 2001.

[144] J. Fridrich, M. Goljan, and D. Hogea. Steganalysis of JPEG images: Breaking the F5 algorithm. In *5th International Workshop on Information Hiding, Noordwijkerhout, The Netherlands, October 7–9, 2002*, volume 2578 of *LNCS*, pages 310–323. Springer, New York, 2002.

[145] J. Fridrich, M. Goljan, D. Hogea, and D. Soukal. Quantitative steganalysis of digital images: Estimating the secret message length. *ACM Multimedia Systems Journal, Special Issue on Multimedia Security*, 9(3):288–302, 2003.

[146] J. Fridrich, M. Goljan, and D. Soukal. Higher-order statistical steganalysis of palette images. In *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents V, Santa Clara, CA, January 21–24, 2003*, pages 178–190, 2003.

[147] J. Fridrich, M. Goljan, and D. Soukal. Perturbed quantization steganography using wet paper codes. In J. Dittman and J. Fridrich, editors, *Proceedings ACM Multimedia and Security Workshop, Magdeburg, Germany, September 20–21, 2004*, pages 4–15. ACM Press, New York, 2004.

[148] J. Fridrich, M. Goljan, and D. Soukal. Perturbed quantization steganography. *ACM Multimedia and Security Journal*, 11(2):98–107, 2005.

[149] J. Fridrich, M. Goljan, and D. Soukal. Steganography via codes for memory with defective cells. In *43rd Conference on Coding, Communication, and Control, September 28–30, 2005*, 2005.

[150] J. Fridrich, M. Goljan, and D. Soukal. Efficient wet paper codes. In M. Barni, editor, *Proceedings, Information Hiding, 7th International Workshop, IH 2005, Barcelona, Spain, June 6–8, 2005*, LNCS. Springer, Berlin, 2006.

[151] J. Fridrich, M. Goljan, D. Soukal, and T. Holotyak. Forensic steganalysis: Determining the stego key in spatial domain steganography. In *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VII, San Jose, CA, January 17–20, 2005*, volume 5681, pages 631–642, 2005.

[152] J. Fridrich and D. Soukal. Matrix embedding for large payloads. *IEEE Trans on Information Security and Forensics*, 1(3):390–394, 2006.

[153] J. Fridrich, R. Du, and M. Long. Steganalysis of LSB encoding in color images. *IEEE International Conference on Multimedia and Expo*, 2000.

[154] G. L. Friedman. The trustworthy camera: Restoring credibility to the photographic image. *IEEE Trans. on Consumer Electronics*, 39(4):905–910, 1993.

[155] G. L. Friedman. Digital camera with apparatus for authentication of images produced from an image file. *United States Patent*, 5,499,294, 1996.

[156] M. S. Fu and O. C. Au. Data hiding for halftone images. In *Proc. SPIE Conf. on Security and Watermarking of Multimedia Data*, volume 3971, pages 228–236, 2000.

[157] T. Furon and P. Duhamel. An asymmetric public detection watermarking technique. In *Proc. of the 3rd Int. Information Hiding Workshop*, pages 88–100, Dresden, Germany, Sept. 1999.

[158] T. Furon and P. Duhamel. Robustness of an asymmetric watermarking technique. *IEEE Int. Conference on Image Processing*, 3:21–24, 2000.

[159] F. Galand and G. Kabatiansky. Information hiding by coverings. In *Proceedings ITW2003, Paris, France, 2003*, pages 151–154, 2003.

[160] R. A. Garcia. Digital watermarking of audio signals using a psychoacoustic auditory model and spread spectrum theory. *Proc. 107th AES Convertion*, 1999.

[161] S. I. Gel'fand and M. S. Pinsker. Coding for channel with random parameters. *Problems of Control and Information Theory*, 9(1):19–31, 1980.

[162] Bernd Girod. What's wrong with mean-squared error? In A. B. Watson, editor, *Digital Images and Human Vision*, chapter 15, pages 207–220. MIT Press, 1993.

[163] R. D. Gitlin, J. F. Hayes, and S. B. Weinstein. *Data Communications Principles*. Plenum Press, 1992.

[164] H. M. Gladney, F. C. Mintzer, and F. Schiattarella. Safeguarding digital library contents and users: Digital images of treasured antiquities. *D-Lib Magazine*, 1997.

[165] M. Goljan, J. Fridrich, and R. Du. Distortion-free data embedding for images. In *4th Int. Information Hiding Workshop*, 2001.

[166] M. Goljan, J. Fridrich, and T. Holotyak. New blind steganalysis and its implications. In E. Delp III and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VIII*, volume 6072, 1–13, 2006.

[167] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley, 1992.

[168] David M. Green and John A. Swets. *Signal Detection Theory and Psychophysics*. Robert E. Krieger Publishing Co., 1973.

[169] J. Haitsma and T. Kalker. A highly robust audio fingerprinting system. In *3rd Int. Conf. on Music Information Retrieval ISMIR 2002*, 2002.

[170] J. Haitsma, T. Kalker, and J. Oostveen. Robust audio hashing for content identification. In *Content Based Multimedia Indexing*, 2001.

[171] C. F. Hall and E. L. Hall. A nonlinear model for the spatial characteristics of the human visual system. *IEEE Trans. Systems Man and Cybernetics*, 7(3):161–170, 1977.

[172] Khaled N. Hamdy, Ahmed H. Tewfik, Ting Chen, and Satoshi Takagi. Time-scale modification of audio signals with combined harmonic and wavelet representations. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 1:439–442, 1997.

[173] A. Hanjalic, G. C. Langelaar, P. M. B. van Roosmalen, J. Biemond, and R. L. Lagendijk. *Image and Video Databases: Restoration, Watermarking and Retrieval*. Elsevier, 2000.

[174] J. J. Harmsen and W. A. Pearlman. Steganalysis of additive noise modelable information hiding. In E. Delp III and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents V, Santa Clara, CA, January 21–24, 2003*, volume 5020, pages 131–142, 2003.

[175] J. J. Harmsen and W. A. Pearlman. Capacity of steganalysis channels. In *Proceedings ACM Multimedia and Security Workshop, New York, NY, August 1–2, 2005*, pages 11–24, 2005.

[176] Fredric J. Harris. On the use of windows for harmonic analysis with the Discrete Fourier Transform. *Proc. IEEE*, 66(1):51–83, 1978.

[177] F. Hartung, P. Eisert, and B. Girod. Digital watermarking of MPEG-4 facial animation parameters. *Comput. and Graphics*, 22(4):425–435, 1998.

[178] F. Hartung and B. Girod. Watermarking of uncompressed and compressed video. *Signal Processing*, 66(3):283–301, 1998.

[179] F. Hartung, J. K. Su, and B. Girod. Spread spectrum watermarking: Malicious attacks and counterattacks. In *SPIE Conf. on Security and Watermarking of Multimedia Content*, volume 3657, pages 147–158, 1999.

[180] Frank Hartung and Martin Kutter. Multimedia watermarking techniques. *Proc. IEEE*, 87(7):1079–1107, 1999.

[181] G. E. Healey and R. Kondepudy. Radiometric CCD camera calibration and noise estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(3):267–276, March 1994.

[182] C. Heegard and A. El-Gamal. On the capacity of computer memory with defects. *IEEE Trans. on Inform. Theory*, 29:731–739, 1983.

[183] H. Heijmans and L. Kamstra. Reversible data embedding based on the Haar wavelet decomposition. In *Proceedings of the Seventh International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 5–13, 2003.

[184] C. W. Helstrom. *Statistical Theory of Signal Detection*. Pergamon Press, 1960.

[185] Emil Frank Hembrooke. Identification of sound and like signals. *United States Patent*, 3,004,104, 1961.

[186] H. Hering, M. Hagmüller, A. Kröpfl, and G. Kubin. Speech watermarking for air traffic control. In *Proceedings of the XII European Signal Processing Conference (EUSIPCO)*, pages 1653–1656, 2004.

[187] H. Hering, M. Hagmüller, and G. Kubin. Safety and security increase for air traffic management through unnoticeable watermark aircraft identification tag transmitted with the VHF voice communication. In *Proceedings of the 22nd Digital Avionics Systems Conference*, pages 4E2,1–10, 2003.

[188] J. J. Hernandez, F. Perez-Gonzalez, J. M. Rodriguez, and G. Nieto. Performance analysis of a 2D multipulse amplitude modulation scheme for data hiding and watermarking still images. *IEEE Journal of Selected Areas in Communication*, 16(4):510–524, 1998.

[189] J. R. Hernandez, M. Amado, and F. Perez-Gonzalez. DCT-domain watermarking techniques for still images: Detector performance analysis and a new structure. *IEEE Transactions on Image Processing*, 9(1):55–68, 2000.

[190] J. R. Hernandez, J.-F. Delaigle, and B. Macq. Improving data hiding by using convolutional codes and soft-decision decoding. In *Proc. SPIE Conf. on Security and Watermarking of Multimedia Content II*, volume 3971, pages 24–47, 2000.

[191] J. R. Hernandez and F. Perez-Gonzalez. Statistical analysis of watermarking schemes for copyright protection of images. *Proceedings of the IEEE*, 87(7):1142–1166, 1999.

[192] Herodotus. *The Histories*. Penguin Books, London, 1996. Translated by Aubrey de Sélincourt.

[193] S. Hetzl and P. Mutzel. A graph–theoretic approach to steganography. In J. Dittmann et al., editor, *Communications and Multimedia Security. 9th IFIP TC-6 TC-11 International Conference, CMS 2005*, volume 3677 of *LNCS*, pages 119–128, Salzburg, Austria, September 19–21, 2005.

[194] K. Hofbauer and H. Hering. Digital signatures for the analogue radio. In *Proceedings of the 5th NASA Integrated Communications Navigation and Surveillance (ICNS) Conference and Workshop*, 2005.

[195] Matthew Holliman and Nasir Memon. Counterfeiting attacks on oblivious block-wise independent invisible watermarking schemes. *IEEE Trans. on Image Processing*, 9(3):432–441, 2000.

[196] G. C. Holst. *CCD Arrays, Cameras, and Displays*, *second edition*, JCD Publishing & SPIE Press, 1998.

[197] L. Holt, B. G. Maufe, and A. Wiener. Encoded marking of a recording signal. *UK Patent*, GB 2196167A, 1988.

[198] C. Honsinger and M. Rabanni. Data embedding using phase dispersion. In *Proc. Int. Conf. on Information Technology*, 2000.

[199] C. W. Honsinger and S. J. Daly. Method for detecting rotation and magnification in images. *United States Patent*, 5,835,639, 1998.

[200] C. W. Honsinger, P. Jones, M. Rabbani, and J. C. Stoffel. Lossless recovery of an original image containing embedded data. *U.S. Patent Application, Docket No. 77102/E-D*, 1999.

[201] N. J. Hopper, J. Langford, and L. von Ahn. Provably secure steganography. In M. Yung, editor, *Advances in Cryptology: CRYPTO 2002*, volume 2442 of *LNCS*, pages 77–92. Springer, 2002.

[202] C.-T. Hsu and J.-L. Wu. Hidden digital watermarks in images. *IEEE Trans. on Image Processing*, 8(1):58–68, 1999.

[203] Y. N. Hsu, H. H. Arsenault, and G. April. Rotation-invariant digital pattern recognition using circular harmonic expansion. *Applied Optics*, 21:4012–4015, 1982.

[204] Dard Hunter. *Handmade Paper and Its Watermarks: A Bibliography*. B. Franklin, New York, 1967.

[205] *Methodology for the subjective assessment of the quality of television pictures—Recommendation ITU-R BT.500-10*. ITU Radiocommunication Assembly, 2000.

[206] D. W. Jacobs. Grouping for recognition. *AI Memo 1177*, MIT, 1989.

[207] J. R. Janesick. *Scientific Charge-Coupled Devices*, volume Monograph PM83. SPIE Press—The International Society for Optical Engineering, January 2001.

[208] N. Jayant, J. Johnston, and R. Safranek. Signal compression based on models of human perception. *Proc IEEE*, 81(10), 1993.

[209] N. Johnson, Z. Duric, and S. Jajodia. *Information Hiding: Steganography and Watermarking—Attacks and Countermeasures*. Kluwer Academic Publishers, 2000.

[210] N. Johnson and S. Jajodia. Steganalysis: The investigation of hidden information. In *Proceedings of the 1998 IEEE Information Technology Conference, Syracuse, NY, September 1–3, 1998*.

[211] N. F. Johnson and S. Jajodia. Steganalysis of images created using current steganography software. In D. Aucsmith, editor, *Information Hiding, 2nd International Workshop, Portland, OR, April 1998*, volume 1525 of *LNCS*, pages 273–289. Springer-Verlag, Berlin, 1998.

[212] N. F. Johnson and S. Jajodia. Steganography: Seeing the unseen. *IEEE Computer*, pages 26–34, February 1998.

[213] JTC1/SC29/WG11 MPEG. Information technology—coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbits/s—part 3: Audio. Technical Report IS11172-3, ISO/IEC, 1992.

[214] David Kahn. *The Codebreakers—The Story of Secret Writing*. Scribner, New York, 1967.

[215] A. B. Kahng, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, Huijuan Wang, and G. Wolfe. Robust IP watermarking methodologies for physical design. *Proceedings 35th Design and Automation Conference*, pages 782–787, 1998.

[216] T. Kalker and A. J. E. M. Janssen. Analysis of watermark detection using SPOMF. In *Proceedings. 1999 International Conference on Image Processing*, volume 1, 1999.

[217] T. Kalker, J.-P. Linnartz, and M. van Dijk. Watermark estimation through detector analysis. In *Proceedings. 1998 International Conference on Image Processing*, volume 1, 1998.

[218] T. Kalker and F. M. Willems. Capacity bounds and code constructions for reversible data-hiding. In E. Delp III, editor, *Proceedings SPIE, Electronic Imaging, Security*

and Watermarking of Multimedia Contents V, volume 5020, pages 604–611, January 2003.

[219] H. R. Kang. *Digital Color Halftoning*. IEEE Press, 1999.

[220] M. Karahan, U. Topkara, M. Atallah, C. Taskiran, E. Lin, and E. Delp. A hierarchical protocol for increasing the stealthiness of steganographic methods. In *Proceedings ACM Multimedia and Security Workshop, Magdeburg, Germany, September 20–21, 2004*, pages 16–24, 2004.

[221] S. Katzenbeisser and F. Petitcolas, editors. *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House, 2000.

[222] S. Katzenbeisser and F. A. P. Petitcolas. On defining security in steganographic systems. In *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IV, San Jose, CA, January 21–24, 2002*, volume 4675, pages 50–56, 2002.

[223] D. H. Kelly. Visual responses to time dependent stimuli. I. Amplitude sensitivity measurements. *J. Opt. Soc. Am. A*, 51:422–429, 1961.

[224] A. Ker. Quantitative evaluation of pairs and RS steganalysis. In E. J. Delp III and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VI, San Jose, CA, January 19–22, 2004*, volume 5306, pages 83–97, 2004.

[225] A. Ker. A general framework for structural analysis of LSB replacement. In M. Barni, et al., editors, *Proceedings 7th Information Hiding Workshop, Barcelona, Spain, June 6–8, 2005*, volume 3727, *Lecture Notes in Computer Science*, pages 276–311. Springer-Verlag, Berlin, 2005.

[226] A. Ker. Improved detection of LSB steganography in grayscale images. In J. Fridrich, editor, *Preproceedings, Information Hiding, 6th International Workshop, IIH 2004, Toronto, Canada, May 23–25, 2004*, volume 3200 of *LNCS*, pages 97–115. Springer-Verlag, Berlin, 2005.

[227] A. Ker. Resampling and the detection of LSB matching in color bitmaps. In E. Delp, et al., editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VII, San Jose, CA, January 16–20, 2005*, volume 5681, pages 1–15, 2005.

[228] A. Ker. Optimally weighted least-squares steganalysis. In E. Delp, et al., editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6502, pages 06-1–06-16, 2007.

[229] A. Ker and R. Boehme. A two-factor error model for quantitative steganalysis. In E. Delp, et al., editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VIII*, volume 6072, pages 06-1–06-16, 2006.

[230] A. D. Ker. Steganalysis of LSB matching in grayscale images. *IEEE Signal Processing Letters*, 12(6):441–444, June 2005.

[231] Aguste Kerckhoff. La cryptographie militaire. *Journal des sciences militaires*, IX(Jan., Feb.):5–38, 161–191, 1883.

[232] Dogan Kesdogan, Jan Egner, and Roland Büschks. Stop-and-Go-MIXes providing probabilistic anonymity in an open system. In *Proc. of the 2nd Int. Information Hiding Workshop*, pages 83–98, 1998.

[233] M. Kharrazi, H. T. Sencar, and N. Memon. Cover selection for steganographic embedding. In *IEEE International Conference on Image Processing*, 2006.

[234] A. Kiayias, Y. Raekow, and A. Russell. Efficient steganography with provable security gurantees. In M. Barni, et al., editors, *Proceedings, Information Hiding, 7th International Workshop, IH Barcelona, Spain, June 6–8, 2005*, volume 3727 of *LNCS*, pages 118–130. Springer-Verlag, Berlin, 2005.

[235] David Kilburn. Dirty linen, dark secrets. *Adweek*, 1997.

[236] J. Kilian, F. T. Leighton, L. R. Matheson, T. Shamoon, and R. E. Tarjan. Resistance of watermarked documents to collusional attacks. Technical Report TR 97-167, NEC Research Institute, 1997.

[237] E. Koch and J. Zhao. Towards robust and hidden image copyright labeling. In *IEEE Workshop on Nonlinear Signal and Image Processing*, 1995.

[238] N. Komatsu and H. Tominaga. Authentication system using concealed image in telematics. *Memoirs of the School of Science and Engineering, Waseda University*, 52:45–60, 1988.

[239] N. Komatsu and H. Tominaga. A proposal on digital watermark in document image communication and its application to realizing a signature. *Electronics and Communications in Japan*, 73(5):208–218, 1990.

[240] D. Kundur and D. Hatzinakos. Blind image deconvolution. *IEEE Signal Processing Magazine*, 13(3):43–64, 1996.

[241] D. Kundur and D. Hatzinakos. Semi-blind image restoration based on telltale watermarking. In *Conference Record of the Thirty-Second Asilomar Conference on Signals, Systems and Computers*, volume 2, 1998.

[242] D. Kundur and D. Hatzinakos. Digital watermarking for telltale tamper proofing and authentication. *Proc. of the IEEE*, 87(7), 1999.

[243] M. Kutter. Watermarking resisting to translation, rotation and scaling. In *Multimedia Systems and Applications*, volume Proc. SPIE-3528:423–431, 1998.

[244] M. Kutter. Performance improvement of spread spectrum based image watermarking schemes through M-ary modulation. In A. Pfitzmann, editor, *Third Int. Workshop on Information Hiding*, volume 1768 of *Lecture Notes in Computer Science*, pages 237–252, 1999.

[245] M. Kutter, S. K. Bhattacharjee, and T. Ebrahimi. Towards second generation watermarking systems. In *IEEE Int. Conf. on Image Processing*, volume 1, pages 320–323, 1999.

[246] M. Kutter and F. Hartung. Introduction to watermarking techniques. In S. Katzenbeisser and F. A. P. Petitcolas, editors, *Information Hiding: Techniques for Steganography and Digital Watermarking*, chapter 5, pages 97–120. Artech House, 2000.

[247] M. Kutter and F. A. P. Petitcolas. A fair benchmark for image watermarking systems. *Security and Watermarking of Multimedia Contents*, Proc. SPIE-3657:226–239, 1999.

[248] M. Kutter, S. Voloshynovskiy, and A. Herrigel. The watermark copy attack. *Security and Watermarking of Multimedia Contents II*, Proc. SPIE-3971:371–380, 2000.

[249] J. Lach, W. Mangione-Smith, and M. Potknojak. Fingerprinting digital circuits on programmable hardware. In *Proc. of the 2nd Int. Information Hiding Workshop*, pages 16–31, 1998.

[250] Jean Laroche and Mark Dolson. Improved phase vocoder time-scale modification of audio. *IEEE Trans. Speech and Audio Proc.*, 7(3):323–332, 1999.

[251] J. Le Moigne. Towards a parallel registration of multiple resolution remote sensing data. In *Proc. of 1995 International Geoscience and Remote Sensing Symposium*, pages 1011–1013, July 1995.

[252] Jaejin Lee and Chee Sun Won. Authentication and correction of digital watermarking images. *Electronic Letters*, 35(11), 1999.

[253] Jaejin Lee and Chee Sun Won. Image integrity and correction using parities of error control coding. In *IEEE Int. Conf. on Multimedia and Expo, 2000*, volume 3, 2000.

[254] K. Lee, D. Kim, T. Kim, and K. Moon, EM estimation of scale factor for quantization-based audio watermarking. *Int. Proc 2nd Workshop on Digital Watermarking*, LCNS 2939: 316–327, 2003.

[255] K. Lee, A. Westfeld, and S. Lee. Category attack for LSB embedding of JPEG images. In *Proceedings of the International Workshop on Digital Watermarking, LNCS Springer-Verlag*, volume 4283, pages 35–38, 2006.

[256] Q. Li and I. J. Cox. Rational dither modulation watermarking using a perceptual model. In *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–4, 2005.

[257] Q. Li and I. J. Cox. Using perceptual models to improve fidelity and provide invariance to valumetric scaling for quantization index modulation watermarking. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 1–4, 2005.

[258] Te-shen Liang and Jeffrey J. Rodriguez. Robust watermarking using robust coefficients. *Security and Watermarking of Multimedia Contents. II*, SPIE-3971:326–335, 2000.

[259] C.-Y. Lin and S.-F. Chang. Semi-fragile watermarking for authenticating JPEG visual content. In *Proc of SPIE, Security and Watermarking in Multimedia Contents II*, volume 3971, 2000.

[260] C.-Y. Lin and Shih-Fu Chang. A robust image authentication algorithm surviving JPEG compression. *Storage and Retireval of Image/Video Databases*, SPIE-, 1998.

[261] C.-Y. Lin and Shih-Fu Chang. Issues and solutions for authenticating MPEG video. *Security amd Watermarking of Multimedia Contents*, Proc. SPIE-3657, 1999.

[262] C.-Y. Lin, M. Wu, J. A. Bloom, I. J. Cox, M. L. Miller, and Y. M. Lui. Rotation, scale and translation resilient public watermarking for images. *Security and Watermarking of Multimedia Contents*, Proc. SPIE-3971:90–98, 2000.

[263] Ching-Yung Lin and Shih-Fu Chang. Distortion modeling and invariant extraction for digital image print-and-scan process. *Int. Symp. on Multimedia Information Processing*, 1999.

[264] E. T. Lin, C. I. Podilchuk, and E. J. Delp. Detection of image alterations using semi-fragile watermarks. In *Proc. SPIE Security and Watermarking of Multimedia Contents III*, volume 3971, 152–163, 2000.

[265] F. Lin and R. D. Brandt. Towards absolute invariants of images under translation, rotation, and dilation. *Pattern Recognition Letters*, 14(5):369–379, 1993.

[266] T. Lindkvist. *Fingerprinting Digital Document*. Ph.D. thesis, Linköping University, 1999.

[267] J.P.M.G. Linnartz and J. C. Talstra. MPEG PTY-marks: Cheap detection of embedded copyright data in DVD-video. In *Proceeding of ESORICS98 5th European Symposium on Research in Computer Security*, 1998.

[268] J.P.M.G. Linnartz, A. C. C. Kalker, and G. F. Depovere. Modelling the false-alarm and missed detection rate for electronic watermarks. In D. Aucsmith, editor, *Workshop on Information Hiding, Portland, OR*, volume 1525 of *Lecture Notes in Computer Science*, pages 329–343. Springer, April, 15–17, 1998.

[269] J.P.M.G. Linnartz and Marten van Dijk. Analysis of the sensitivity attack against electronic watermarks in images. In *Workshop on Information Hiding, Portland, OR*, April, 15–17, 1998.

[270] J. Löfvenberg. *Random Codes for Digital Fingerprinting*. Ph.D. thesis, Linköping University, 1999.

[271] S. Low and N. Maxemchuk. Performance comparison of two text marking methods. *IEEE Journal of Selected Areas in Communication*, 16:561–572, 1998.

[272] Chun-Shien Lu, Hong-Yuan Mark Liao, Shih-Kun Huang, and Chwen-Jye Sze. Cocktail watermarking on images. In A. Pfitzmann, editor, *Third Int. Workshop on Information Hiding*, volume 1768 of *Lecture Notes in Computer Science*, pages 333–347, 1999.

[273] P. Lu, X. Luo, Q. Tang, and L. Shen. An improved sample pairs method for detection of LSB embedding. In J. Fridrich, editor, *Information Hiding, 6th International Workshop*, volume 3200 of *LNCS*, pages 116–127. Springer-Verlag, Berlin, 2004.

[274] J. Lubin. The use of psychophysical data and models in the analysis of display system performance. In A. B. Watson, editor, *Digital Images and Human Vision*, pages 163–178. MIT Press, 1993.

[275] M. Luby. LT codes. In *Proceedings, 43rd Annual IEEE Symposium on Foundations of Computer Science, November 16–19, 2002*, pages 271–282, 2002.

[276] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. *Proc. International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[277] N. B. Lucena, S. J. Chapin, J. Pease, and P. Yadollahpour. Syntax and semantics—preserving application-layer protocol steganography. In J. Fridrich, editor, *Information Hiding, 6th International Workshop, Toronto, Canada, May 23–25, 2004*, volume 3200 of *LNCS*, pages 164–179, 2004.

[278] J. Lukš, J. Fridrich, and M. Goljan. Digital camera identification from sensor noise. *IEEE Transactions on Information Security and Forensics*, 1(2), 2006.

[279] B. M. Macq and J-J. Quisquater. Cryptology for digital TV broadcasting. *Proc. of the IEEE*, 83(6):944–957, 1995.

[280] F. J. MacWilliams and Neil J. A. Sloane. Pseudo-random sequences and arrays. *Proc. of the IEEE*, 64(12):1715–1729, 1976.

[281] M.J.J.J.B. Maes and C.W.A.M. van Overveld. Digital watermarking by geometric warping. In *IEEE Int. Conf. on Image Processing*, volume 2, pages 424–426, 1998.

[282] J. L. Mannos and J. J. Sakrison. The effects of a visual fidelity criterion on the encoding of images. *IEEE Trans. Inform. Theory*, IT-4:525–536, 1974.

[283] Hedy Kiesler Markey and George Antheil. Secret communication system. Technical Report 2,292,387, *United States Patent*, 1942.

[284] L. Marks. *Between Silk and Cyanide: A Codemaker's War, 1941–1945*. Free Press, 2000.

[285] Mathison. *Gentlemen's Magazine*, XLIX, 1779.

[286] J. Meng and S.-F. Chang. Embedding visible video watermarks in the compressed domain. In *IEEE Int. Conf. on Image Processing*, volume 1, pages 474–477, 1998.

[287] M. K. Mihcak, I. Kozintsev, K. Ramchandran, and P. Moulin. Low-complexity image denoising based on statistical modeling of wavelet coefficients. *IEEE Signal Processing Letters*, 6(12):300–303, December 1999.

[288] G. L. Miller, G. J. Doerr, and I. J. Cox. Dirty-paper trellis codes for watermarking. In *IEEE Int. Conf. on Image Processing*, volume 2, pages II-129–II-132, 2002.

[289] M. L. Miller and J. A. Bloom. Computing the probability of false watermark detection. In *Proceedings of the Third International Workshop on Information Hiding*, 1999.

[290] M. L. Miller, I. J. Cox, and J. A. Bloom. Informed embedding: Exploiting image and detector information during watermark insertion. In *IEEE International Conference on Image Processing*, September 2000.

[291] M. L. Miller, G. Doerr, and I. J. Cox. Applying informed coding and informed embedding to design a robust high capacity watermark. *IEEE Trans. on Image Processing*, 13(6), 2004.

[292] M. L. Miller, M. A. Rodriguez, and I. J. Cox. Audio fingerprinting: Nearest neighbor search in high-dimensional binary spaces. In *IEEE Multimedia Signal Processing Workshop*, pages 182–185, 2002.

[293] M. L. Miller, M. A. Rodriguez, and I. J. Cox. Audio fingerprinting: Nearest neighbor search in high-dimensional binary spaces. *J. of VLSI Signal Processing*, 2005.

[294] F. Mintzer, J. Lotspiech, and N. Morimoto. Safeguarding digital library contents and users: Digital watermarking. *D-Lib Magazine*, http://www.dlib.org/dlib/december97/ibm/12lotspiech.html#FM2, 1997.

[295] S. P. Mohanty, K. R. Ramakrishnan, and M. S. Kankanhalli. A DCT domain visible watermarking technique for images. In *IEEE Int. Conf. on Multimedia and Expo*, volume 2, pages 1029–1032, 2000.

[296] Warren D. Moon, Richard J. Weiner, Robert A. Hansen, and Robert N. Linde. Broadcast signal identification system. *United States Patent*, 3,919,479, 1975.

[297] T. Moriya, Y. Takashima, T. Nakamura, and N. Iwakami. Digital watermarking schemes based on vector quantization. In *Proc. of IEEE Workshop on Speech Coding for Telecommunications*, pages 95–96, 1997.

[298] Ira S. Moskowitz and Li Wu Chang. An entropy-based framework for database inference. In *Proc. of the 3rd Int. Information Hiding Workshop*, 1999.

[299] P. Moulin and R. Koctter. Data-hiding. *Proc. IEEE*, 93(12), 2083–2127, 2005.

[300] P. Moulin, M. K. Mihcak, and G. I. Lin. An information-theoretic model for image watermarking and data hiding. In *Proceedings ICIP'00, IEEE International Conference on Image Processing, Vancouver, Canada, September 2000*, 2000.

[301] P. Moulin and J. A. O'Sullivan. Information-theoretic analysis of information hiding. *Preprint*, available from http://www.ifp.uiuc.edu/moulin/Papers, 1999.

[302] P. Moulin, A. K. Cŏtetr, and Rkočllcr. "Optimal" spare-QIM codes zero-rate blind watermarking. *Proceedings of ICASSP*, volume 3, 73–76, 2004.

[303] P. Moulin and J. A. O'Sullivan. Information-theoretic analysis of information hiding. *IEEE Trans. Inform. Theory*, 49(3):563–593, March 2003.

[304] P. Moulin and Y. Wang. New results on steganographic capacity. In *Proceedings CISS Conference, Princeton, NJ, March 2004*, 2004.

[305] F. Müller. Distribution shape of two-dimensional DCT coefficients of natural images. *Electronics Letters*, 29(22):1935–1936, 1993.

[306] H. Murase and S. Nayar. Visual learning and recognition of 3D objects from appearance. *Int. J. Computer Vision*, 14(1):5–25, 1995.

[307] H. Muratani. A collusion-secure fingerprinting code reduced by Chinese remaindering and its random-error resilience. In *Proceedings of Int. Workshop on Information Hiding '01*, 2001.

[308] National Institute of Standards and Technology. Digital signature standard. Technical Report NIST FIPS PUB 185, U.S. Department of Commerce, May 1994.

[309] H. Neuschmied, H. Mayer, and E. Battle. Content-based indentification of audio titles on the Internet. In *Proc. Int. Conf. on Web Delivering of Music*, 2001.

[310] Z. Ni, Y. Q. Shi, N. Ansari, and W. Su. Reversible data hiding. *IEEE Transactions on Circuits Systems and Video Technology*, 16(3):354–362, March 2006.

[311] H. Noda, M. Niimi, and E. Kawaguchi. Application of QIM with dead zone for histogram preserving JPEG steganography. In *Proceedings ICIP, Genova, 2005*, 2005.

[312] R. Ohbochi, H. Masuda, and M. Aono. Watermarking three-dimensional polygonal models through geometric and topological modifications. *IEEE Journal of Selected Areas in Communication*, 16(4):551–560, 1998.

[313] J. Oostveen, T. Kalker, and J. Haitsma. Feature extraction and a database strategy for video fingerprinting. In *5th Int. Conf. on Visual Information Systems*, volume LNCS 2314, pages 117–128. Springer-Verlag, 2002.

[314] J. C. Oostveen, A. A. C. Kalker, and M. Staring. Adaptive quantization watermarking. In *Proc. of SPIE: Security, Steganography, and Watermarking of Multimedia Contents VI*, volume 5306, pages 37–39, San Jose, CA. January 2004.

[315] Alan V. Oppenheim and Ronald W. Schafer. *Digital Signal Processing*. Prentice-Hall, 1975.

[316] J. J. K. Ó Ruanaidh, E. M. Boland, and W. J. Dowling. Phase watermarking of digital images. *Proc. IEEE Int. Conf. on Image Processing*, III:239–242, 1996.

[317] J. J. K. Ó Ruanaidh and T. Pun. Rotation, scale and translation invariant spread spectrum digital image watermarking. *Signal Processing*, 66(3):303–317, 1998.

[318] Ted Painter and Andreas Spanias. Perceptual coding of digital audio. *Proc. IEEE*, 88(4):451–513, 2000.

[319] H. C. Papadopoulos and C.-E.W. Sundberg. Simultaneous broadcasting of analog FM and digital audio signals by means of precanceling techniques. In *IEEE Int. Conf. on Communications, ICC '98*, volume 2, pages 728–732, 1998.

[320] Athanasios Papoulis. *Probability, Random Variables, and Stochastic Processes*, *third edition*. McGraw-Hill, 1991.

[321] A. Patrizio. Why the dvd hack was a cinch. *Wired*, November 1999.

[322] W. B. Pennebaker and J. L. Mitchell. *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, 1993.

[323] S. Pereira, J. J. K. Ó Ruanaidh, and T. Pun. Secure robust digital watermarking using the lapped othogonal transform. In *SPIE Conf. on Security and Watermarking of Multimedia Content*, volume SPIE 3657, pages 21–30, 1999.

[324] S. Pereira and T. Pun. Robust template matching for affine resistant image watermarks. *IEEE Transactions on Image Processing*, 9(6):1123–1129, 2000.

[325] S. Pereira, J. J. K. Ó Ruanaidh, F. Deguillaume, G. Csurka, and T. Pun. Template based recovery of Fourier-based watermarks using log-polar and log-log maps. In *IEEE International Conference on Multimedia Computing and Systems*, volume 1, pages 870–874, 1999.

[326] S. Pereira, S. Voloshynovskiy, and T. Pun. Effective channel coding for DCT watermarks. In *IEEE Int. Conf. on Image Processing*, volume 3, pages 671–673, 2000.

[327] F. Perez-Gonzalez, M. Barni, A. Abrardo, and C. Mosquera. Rational dither modulation: A novel data-hiding method robust to value-metric scaling attacks. In *IEEE International Workshop on Multimedia Signal Processing*, Siena, Italy, Sept. 2004.

[328] Fabian A. P. Petitcolas, Ross J. Anderson, and Markus G. Kuhn. Attacks on copyright marking systems. In *Workshop on Information Hiding, Portland, OR*, April, 15–17, 1998.

[329] Fabien A. P. Petitcolas, Ross Anderson, and Markus G. Kuhn. Information hiding—a survey. *Proc. IEEE*, 87(7):1062–1077, 1999.

[330] T. Pevný and J. Fridrich. Multiclass blind steganalysis for JPEG images. In E. Delp III and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VIII*, 6072, 01–013, 2006.

[331] T. Pevný and J. Fridrich. Merging Markov and DCT features for multi-class JPEG steganalysis. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505, pages 03-1–03-13, January 2007.

[332] R. L. Pickholtz, D. L. Schilling, and L. B. Milstein. Theory of spread-spectrum communications—a tutorial. *IEEE Trans. on Communications*, 30(5):855–884, 1982.

[333] S. Pietrobon, R. Deng, A. Lafanechere, G. Ungerboeck, and D. Costello. Trellis-coded multidimensional phase modulation. *IEEE Transactions on Information Theory*, 36:6389, 1990.

[334] *Playboy* magazine. Playboy Enterprises International, Inc., November 1972.

[335] Christine I. Podilchuk and Wenjun Zeng. Image-adaptive watermarking using visual models. *IEEE Journal of Selected Areas in Communication*, 16(4):525–539, 1998.

[336] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[337] R. D. Preuss, S. E. Roukos, A. W. F. Huggins, H. Gish, M. A. Bergamo, P. M. Peterson, and Derr A. G. Embedded signalling. *United States Patent*, 5,319,735, 1994.

[338] N. Provos. Defending against statistical steganalysis. In *10th USENIX Security Symposium, Washington, DC*, 2001.

[339] N. Provos and P. Honeyman. Detecting steganographic content on the Internet. Technical Report 01-11, CITI, August 2001.

[340] L. Qiao and K. Nahrstedt. Non-invertible watermarking methods for MPEG encoded audio. In *Proc. SPIE Conf. on Security and Watermarking of Multimedia Data*, volume 3657, pages 194–203, 1999.

[341] R. Radhakrishnan, M. Kharrazi, and N. Memon. Data masking a new approach for data hiding? *Journal of VLSI Signal Processing Systems*, 41(3):293–303, November 2005.

[342] H. A. Rahmel. System for determining the listen habits of wave signal receiver users. *United States Patent*, 2,513,360, 1950.

[343] M. Ramkumar. *Data Hiding in Multimedia: Theory and Applications*. Ph.D. thesis, New Jersey Institute of Technology, 1999.

[344] Chris Reidy. Knocking off the knockoff. *The Boston Globe*, page D1, November 17, 2000.

[345] R. C. Reininger and J. D. Gibson. Distribution of the two-dimensional DCT coefficients for images. *IEEE Trans. on Comm.*, 31(6):835–839, 1983.

[346] Christian Rey and Jean-Luc Dugelay. Blind detection of malicious alterations on still images using robust watermarks. *IEE Seminar: Secure Images and Image Authentication*, pages 7/1–7/6, 2000.

[347] Geoffrey B. Rhoads. Image steganography system featuring perceptually adaptive and globally scalable signal embedding. *United States Patent*, 5,748,763, 1998.

[348] Geoffrey B. Rhoads. Steganography system. *United States Patent*, 5,850,481, 1998.

[349] R. L. Rivest. RFC 1321: The MD5 Message-Digest Algorithm. *Internet Activities Board*, 1992.

[350] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptsystems. *Communications of the ACM*, 21(2):120–126, 1978.

[351] P. M. J. Rongen, M.J.J.J.B. Maes, and C.W.A.M. van Overveld. Digital image watermarking by salient point modification practical results. In *SPIE Conf. on Security and Watermarking of Multimedia Content*, volume 3657, pages 273–282, 1999.

[352] C. B. Rorabaugh. *Error Coding Cookbook: Practical C/C++ Routines and Recipes for Error Detection and Correction*. McGraw-Hill, 1996.

[353] H. L. Royden. *Real Analysis*. MacMillan Publishing Co., 1968.

[354] D. L. Ruderman and W. Bialek. Statistics of natural images: Scaling in the woods. *Physics Review Letters*, 73:814–817, 1994.

[355] J. Ryan. Method and apparatus for preventing the copying of a video program. *United States Patent*, 4,907,093, 1990.

[356] Screen Actors Guild. http://www.sag.org/pressreleases/prla990826spotchecks.html, 1999.

[357] P. Sallee. Model-based steganography. In T. Kalker, I. J. Cox, and Y. M. Ro, editors, *Digital Watermarking, 2nd International Workshop, IWDW 2003, Seoul, Korea, October 20–20, 2003*, volume 2939 of *LNCS*, pages 154–167. Springer-Verlag, New York, 2004.

[358] D. V. Sarwate and M.-B. Pursley. Cross-correlation properties of pseudo-random and related sequences. *Proc. of the IEEE*, 688(5):593–619, 1980.

[359] K. Sayood. *Introduction to Data Compression, second edition*. Morgan Kaufmann, 2000.

[360] E. Sayrol, J. Vidal, S. Cabanillas, and S. Santamaria. Optimum watermark detection for color images. In *IEEE Int. Conf. on Image Processing*, volume 2, pages 231–235, 1999.

[361] M. Schneider and S.-F. Chang. A robust content based digital signature for image authentication. In *IEEE Int. Conf. on Image Processing*, volume 3, pages 227–230, 1996.

[362] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, 1996.

[363] Leonard Schuchman. Dither signals and their effect on quantization noise. *IEEE Trans. Comm. Tech.*, 12:162–165, 1964.

[364] SDMI Portable Device Specification—Part 1, Version 1.0. Technical Report pdwg99070802, Secure Digital Music Initiative, 1999.

[365] C. E. Shannon. Communications theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, October 1949.

[366] C. E. Shannon. Channels with side information at the transmitter. *IBM Journal of Research and Development*, pages 289–293, 1958.

[367] T. Sharp. An implementation of key-based digital signal steganography. In I. S. Moskowitz, editor, *Information Hiding, 4th International Workshop, IHW 2001, Pittsburgh, PA, April 25–27, 2001*, volume 2137 of *LNCS*, pages 13–26. Springer-Verlag, New York, 2001.

[368] Y. Sheng and H. H. Arsenault. Experiments on pattern recognition using invariant Fourier-Mellin descriptors. *J. Opt. Soc. Am. A*, 3:771–776, 1986.

[369] Y. Sheng and J. Duvernoy. Circular-Fourier–Radial-Mellin transform descriptors for pattern recognition. *J. Opt. Soc. Am. A*, 3:885–888, 1986.

[370] Y. Q. Shi, C. Chen, and W. Chen. A Markov process based approach to effective attacking JPEG steganography. In N. Johnson and J. Camenisch, editors, *Information Hiding, 8th International Workshop*, volume 4437 of *Lecture Notes in Computer Science*. Springer-Verlag, New York, 2006.

[371] I. D. Shterev and R. L. Legendijk, Amplitude scale estimation for quantization-based water marking. *IEEE Trans. on Signal Processing*, 54(11):4146–4155, 2006.

[372] I. D. Shterev, R. L. Legendijk, and R. Hesdens, Statistical amplitude scale estmation for quantization-based watermarking. *Proc. SPIE Security, Steganography and Watermarking of Multimedia Contents VI*, volume 5306, 2004.

[373] M. Sidorov. Hidden markov models and steganalysis. In *Proceedings ACM Multimedia and Security Workshop, Magdeburg, Germany, September 20–21, 2004*, pages 63–67, 2004.

[374] G. Simmons. A Survey of information authentication. *Contemporary Cyptography, The Science of Information Integrity*. IEEE Press, 1992.

[375] Gustavus J. Simmons. The prisoner's problem and the subliminal channel. In D. Chaum, editor, *Advances in Cryptology—CRYPTO 83*, pages 51–67. Plenum Press, 1983.

[376] Gustavus J. Simmons. The history of subliminal channels. In Ross Anderson, editor, *Information Hiding: First International Workshop*, volume 1174 of *Lecture Notes in Computer Science*, pages 237–256. Springer, 1996.

[377] Eero P. Simoncelli. Statistical models for images: Compression, restoration and synthesis. *Thirty-First Asilomar Conf. on Signals, Systems, and Computers*, pages 673–678, 1997.

[378] J. Simpson and E. Weiner, editors. *Oxford English Dictionary*. Oxford University Press, 2000.

[379] W. D. Smith. *Studies in Computational Geometry Motivated by Mesh Generation*. Ph.D. thesis, Princeton University, 1988.

[380] V. Solachidis, N. Nikolaidis, and I. Pitas. Watermarking polygonal lines using fourier descriptors. *IEEE Int. Conf. on Image Processing*, 4:1955–1958, 2000.

[381] K. Solanki, K. Sullivan, U. Madhow, B. S. Manjunath, and S. Chandrasekaran. Provably secure steganography: Achieving zero K-L divergence using statistical restoration. In *Proceedings ICIP, Atlanta, GA*, October 2006.

[382] D. Soukal, J. Fridrich, and M. Goljan. Maximum likelihood estimation of secret message length embedded using PMK steganography in spatial domain. In *Proceedings*

SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VII, San Jose, CA, January 16–20, 2005, volume 5681, pages 595–606, 2005.

[383] M. R. Spiegel. *Schaum's Outline of Theory and Problems of Statistics, third edition*. McGraw-Hill, New York, 1961.

[384] J. Stach and A. M. Alattar. A high-capacity invertible data-hiding algorithm using a generalized reversible integer transform. In E. Delp III, editor, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VI*, volume 5306, pages 386–396, January 2003.

[385] J. P. Stern, G. Hachez, F. Koeune, and J. J. Quisquater. Robust object watermarking: Application to code. In *Proceedings of Info Hiding '99*, volume 1768, *Lecture Notes in Computer Science*, pages 368–378, 1999.

[386] G. W. W. Stevens. *Microphotography and Photofabrication at Extreme Resolutions*. Chapman&Hall, London, 1968.

[387] D. R. Stinson. *Cryptography: Theory and Practice*. CRC Press, 1995.

[388] T. G. Stockham. Image processing in the context of a visual model. *Proc. IEEE*, 60(7):828–841, 1972.

[389] H. S. Stone. Analysis of attacks on image watermarks with randomized coefficients. Technical Report TR 96-045, NEC Research Institute, 1996.

[390] H. S. Stone. Progressive wavelet correlation using Fourier methods. *IEEE Transactions on Signal Processing*, 47(1):97–107, Jan. 1999.

[391] D. Storck. A new approach to integrity of digital images. In *IFIP Conf. on Mobile Communication*, 1996.

[392] J. K. Su and B. Girod. On the imperceptibility and robustness of digital fingerprints. *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, 2:530–535, 1999.

[393] J. K. Su and B. Girod. Power-spectrum condition for energy-efficient watermarking. In *IEEE Int. Conf. on Image Processing*, volume 1, pages 301–305, 1999.

[394] K. Sullivan, U. Madhow, B. S. Manjunath, and S. Chandrasekaran. Steganalysis for Markov cover data with applications to images. *In IEEE Transactions on Information Security and Forensics*, 1(2): 275–287, 2006.

[395] M. D. Swanson, B. Zhu, and A. H. Tewfik. Transparent robust image watermarking. In *Proc. IEEE Int. Conf. on Image Processing*, volume 3, pages 211–214, 1996.

[396] M. D. Swanson, B. Zhu, and A. H. Tewfik. Data hiding for video-in-video. In *Proc. IEEE Int. Conf. on Image Processing*, volume 2, pages 676–679, 1997.

[397] M. D. Swanson, B. Zhu, A. H. Tewfik, and L. Boney. Robust audio watermarking using perceptual masking. *Signal Processing*, 66(3):337–355, 1998.

[398] W. Szepanski. A signal theoretic method for creating forgery-proof documents for automatic verification. In J. S. Jackson, editor, *1979 Carnahan Conf. on Crime Countermeasures*, pages 101–109, 1979.

[399] A. Tacticius. *How to Survive Under Siege/Aineas the Tactician*. Clarendon ancient history series, 1990.

[400] K. Tanaka, Y. Nakamura, and K. Matsui. In *Military Communications Conference, 1990. MILCOM '90, Conference Record, A New Era. 1990 IEEE*, volume 1, pages 216–220, 1990.

[401] G. Tardos, Optimal probabilistic fingerprint codes. *Annual ACM Symposium on Theory of Computing STOC*, pages 116–125, 2003.

[402] J. Taylor. *DVD Demystified*. McGraw-Hill, 1998.

[403] M. M. Taylor. Visual discrimination and orientation. *J. Opt. Soc. Am. A*, 53:763–765, 1963.

[404] A. Tefas and I. Pitas. Multi-bit image watermarking robust to geometric distortions. In *IEEE Int. Conf. on Image Processing*, volume 3, pages 710–713, 2000.

[405] Telecommunications Act of 1996, Implementation of Section 305.

[406] Telecommunications Act of 1996, United States Public Law 104-104.

[407] E. Terhardt. Calculating virtual pitch. *Hearing Research*, 1:155–182, 1979.

[408] J. Tian. Wavelet-based reversible date-embedding using a difference expansion. *IEEE Transactions on Circuit Systems and Video Technology*, 13(8):890–896, August 2003.

[409] A. Z. Tirkel, C. F. Osbourne, and T. E. Hall. Image and watermark registration. *Signal Processing*, 66(3):373–383, 1998.

[410] W. Trappe, M. Wu, Z. J. Wang and K. J. R. Liu. Anti-collusion fingerprinting for multimedia. *IEEE Transactions Signal Processing*, 51(4):1069–1087, 2003.

[411] William M. Tomberlin, Louis G. MacKenzie, and Paul K. Bennett. System for transmitting and receiving coded entertainment programs. *United States Patent*, 2,630,525, 1953.

[412] B. S. Tsybakov. Defect and error correction. *Probl. Pered. Inform.*, 11:21–30, July–September 1975. Translated from Russian.

[413] R. Tzschoppe, R. Buml, J. B. Huber, and A. Kaup. Steganographic system based on higher-order statistics. In E. Delp, et al., editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents V, 2003*, volume 5020, pages 156–166, 2003.

[414] S. Ullman and R. Basri. Recognition by linear combinations of models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(10):992–1006, 1991.

[415] G. Ungerboeck. Channel coding with multilevel/phase signals. *IEEE Transactions on Information Theory*, IT-28:55–67, 1982.

[416] United States Code, Title 17.

[417] Christian J. van den Branden Lambrecht and Joyce E. Farrell. Perceptual quality metric for digitally coded color images. *Proc. EUSIPCO*, pages 1175–1178, 1996.

[418] M. van der Veen, A. van Leest, and F. Bruekers. High capacity reversible watermarking for audio. In E. Delp, et al., editors, *Proceedings SPIE, Electronic Imaging, Security and Watermarking of Multimedia Contents V*, volume 5020, pages 1–11, January 2003.

[419] M. van Dijik and F. Willems. Embedding information in grayscale images. In *Proceedings of the 22nd Symposium on Information and Communication Theory in the Benelux, Enschede, The Netherlands, May 15–16, 2001*, pages 147–154, 2001.

[420] A. van Leest, M. van der Veen, and F. Bruekers. Reversible watermarking for images. In E. Delp, et al., editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VI*, volume 5306, pages 374–385, January 2003.

[421] R. G. van Schyndel, A. Z. Tirkel, and C. F. Osborne. A digital watermark. In *Int. Conf. on Image Processing*, volume 2, pages 86–90. IEEE, 1994.

[422] R. G. van Schyndel, A. Z. Tirkel, and I. D. Svalbe. Key independent watermark detection. In *Proc. International Conference on Multimedia Computing and Systems*, pages 580–585. IEEE, 1999.

[423] R. G. van Schyndel, A. Z. Tirkel, I. D. Svalbe, T. E. Hall, and C. F. Osborne. Algebraic construction of a new class of quasi-orthogonal arrays in steganography. In *Proc. of SPIE on Security and Watermarking of Multimedia Contents*, volume 3657, pages 354–364, 1999.

[424] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[425] Paul Viola and William M. Wells III. Alignment by maximization of mutual information. *Int. J. Computer Vision*, 24(2):137–154, 1997.

[426] B. Vasudev and V. Ratnakar, Fragile watermarks for detecting tampering in images. *United States Patent*, 6,064,764, 2000.

[427] A. J. Viterbi. *CDMA: Principles of Spread Spectrum Communications*. Addison Wesley Longman Inc., 1995.

[428] C. D. Vleeschouwer, J. F. Delaigle, and B. Macq. Circular interpretation of bijective transformations in lossless watermarking for media asset management. *IEEE Transactions on Multimedia*, 5(1):97–105, March 2003.

[429] S. Voloshynovskiy, A. Herrigel, N. Baumgaetner, and T. Pun. A stochastic approach to content adaptive digital image watermarking. In *Third International Workshop on Information Hiding*, 1999.

[430] L. von Ahn and N. Hopper. Public-key steganography. In *Advances in Cryptology: Proceedings of EUROCRYPT, Interlaken, Switzerland, May 2–6*, volume 3027 of *LNCS*, pages 323–341. Springer-Verlag, Heidelberg, 2004.

[431] Martin J. Wainwright and Eero P. Simoncelli. Scale mixtures of Gaussians and the statistics of natural images. In S. A. Solla, T. K. Leen, and K. R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 855–861. MIT Press, 2000.

[432] C. Walker. Personal communication, June 2000.

[433] C. Wang, G. Doerr, and I. J. Cox. Trellis coded modulation to improve dirty paper trellis watermarking. In *SPIE Proceedings on Security, Steganography and Watermarking of Multimedia Contents IX*, 2007.

[434] S.-G. Wang and K. T. Knox. Embedding digital watermarks in halftone screens. In *Proc. SPIE Conf. on Security and Watermarking of Multimedia Data*, volume 3971, pages 218–227, 2000.

[435] X. Wang, S. Chen, and S. Jajodia. Tracking anonymous peer-to-peer VoIP calls on the Internet. In *12th ACM Conf. on Computer and Communications Security*, pages 81–91, 2005.

[436] Y. Wang and P. Moulin. Steganalysis of block-structured stegotext. In *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Content VI, San Jose, CA, January 19–22, 2004*, volume 5306, pages 477–488, 2004.

[437] A. B. Watson, G. Y. Yang, J. A. Solomon, and J. Villasenor. Visual thresholds for wavelet quantization error. *Human Vision and Electronic Imaging*, SPIE-2657:381–392, 1996.

[438] Andrew B. Watson. DCT quantization matrices optimized for individual images. *Human Vision, Visual Processing, and Digital Display IV*, SPIE-1913:202–216, 1993.

[439] Andrew B. Watson, editor. *Digital Images and Human Vision*. MIT Press, 1993.

[440] P. Wayner. Mimic functions. *CRYPTOLOGIA*, 16(3):193–214, July 1992.

[441] P. Wayner. *Disappearing Cryptology*, *second edition*. Morgan Kaufmann, San Francisco, 2002.

[442] A. Westfeld. High capacity despite better steganalysis (F5—a steganographic algorithm). In I. S. Moskowitz, editor, *Information Hiding, 4th International Workshop*, volume 2137 of *LNCS*, pages 289–302. Springer-Verlag, New York, 2001.

[443] A. Westfeld. Detecting low embedding rates. In F. A. P. Petitcolas, editor, *Information Hiding, 5th International Workshop, IH 2002, Noordwijkerhout, The Netherlands, October 7–9, 2002*, volume 2578 of *LNCS*, pages 324–339. Springer-Verlag, Berlin, 2002.

[444] A. Westfeld and A. Pfitzmann. Attacks on steganographic systems. In A. Pfitzmann, editor, *Information Hiding, 3rd International Workshop, IH'99, Dresden, Germany, September 29–October 1, 1999*, volume 1768 of *LNCS*, pages 61–75. Springer-Verlag, New York, 2000.

[445] B. Widrow. Statistical analysis of amplitude-quantized sample-data systems. *Trans. AIEE (Applications and Industry)*, 79:555–568, 1960 (Jan. 1961 section).

[446] E. H. Wilkins. *A History of Italian Literature*. Oxford University Press, London, 1954.

[447] Raymond B. Wolfgang and Edward J. Delp. A watermark for digital images. In *Proceedings of the 1996 International Conference on Image Processing*, volume 3, pages 219–222, 1996.

[448] Raymond B. Wolfgang, C. I. Podilchuk, and E. J. Delp. Perceptual watermarks for digital images and video. *Proc. of the IEEE*, 87(7):1108–1126, 1999.

[449] Ping Wah Wong. A public key watermark for image verification and authentication. In *IEEE Int. Conf. on Image Processing*, volume 1, 1998.

[450] Ping Wah Wong and Edward J. Delp, editors. *Security and Watermarking of Multimedia Contents*, volume 3657. Society of Photo-optical Instrumentation Engineers, 1999.

[451] Ping Wah Wong and Edward J. Delp, editors. *Security and Watermarking of Multimedia Contents II*, volume 3971. Society of Photo-optical Instrumentation Engineers, 2000.

[452] Chung-Ping Wu, Po-Chyi Su, and C.-C. Jay Kuo. Robust and efficient digital audio watermarking using audio content analysis. *Security and Watermarking of Multimedia Contents*, Proc. SPIE-3971:382–392, 2000.

[453] M. Wu, E. Tang, and B. Liu. Data hiding in digital binary images. *IEEE International Conference on Multimedia & Expo (ICME)*, 2000.

[454] Min Wu and Bede Liu. Watermarking for image authentication. In *IEEE Int. Conf. on Image Processing*, volume 2, pages 437–441, 1998.

[455] R. Wu and H. Stark. Rotation-invarint pattern recognition suing optimum feature extraction. *Applied Optics*, 24:179–184, 1985.

[456] A. D. Wyner and J. Ziff. The rate distortion function for source coding with side information at the decoder. *IEEE Trans. Inform. Theory*, 22:1–10, 1976.

[457] Changsheng Xu, Jiankang Wu, and Qibin Sun. Audio registration and its application to digital watermarking. *Security and Watermarking of Multimedia Contents*, Proc. SPIE-3971:393–401, 2000.

[458] G. Xuan, Y. Q. Shi, J. Gao, D. Zou, C. Yang, Z. Zhang, P. Chai, C. Chen, and W. Chen. Steganalysis based on multiple features formed by statistical moments of wavelet characteristic functions. In *Proceedings 7th Information Hiding Workshop, Barcelona, Spain, June 6–8, 2005*, 2005.

[459] G. Xuan, Q. Yao, C. Yang, J. Gao, P. Chai, Y. Q. Shi, and Z. Ni. Lossless data hiding using histogram shifting method based on integer wavelets. *Proc. Int. Workshop on Digital Watermarking,* volume 4283, pages 323–332, November 2006.

[460] W. S. E. Yang and W. Sun. On information embedding when watermarks and covertexts are correlated. In *IEEE International Symposium on Information Theory*, pages 346–350, 2006.

[461] Nakajima Yasumasa, Ichihara Shintaro, and Mogami Kazuto. Digital camera and image authentication system using the same. *Japanese Patent*, JP 11215452, 1999.

[462] B.-L. Yeo and M. M. Yeung. Watermarking 3D objects for verification. *IEEE Computer Graphics and Applications*, 19(1):36–45, 1999.

[463] M. Yeung and F. Mintzer. An invisible watermarking technique for image verification. In *Proc. Int. Conf. Image Processing*, volume 1, pages 680–683, 1997.

[464] Gwo-Jong Yu, Chun-Shien Lu, H. Y. M. Liao, and Jang-Ping Sheu. Mean quantization blind watermarking for image authentication. In *IEEE Int. Conf. on Image Processing*, volume 3, pages 706–709, 2000.

[465] X. Yu, Y. Wang, and T. Tan. On estimation of secret message length in J-steg-like steganography. In *Proceedings, International Conference on Pattern Recognition, Cambridge, UK, August 23–26, 2004*, volume 4, pages 673–676, 2004.

[466] R. Zamir, S. Shamai, and U. Erez. Nested linear/lattice codes for structured multiterminal binning. *IEEE Transactions on Information Theory*, 48(6):1250–1276, 2002.

[467] Q. Zhang and N. Boston. Quantization index modulation using E_8 lattice. In *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, volume 41, pages 488–489, 2003.

[468] T. Zhang and X. Ping. A fast and effective steganalytic technique against jsteg-like algorithms. In *Proceedings, Symposium on Applied Computing, Melbourne, FL*, pages 307–311, 2003.

[469] T. Zhang and X. Ping. A new approach to reliable detection of LSB steganography in natural images. *Signal Processing*, 83(10):2085–2094, 2003.

[470] Bin Zhu and A. H. Tewfik. Media compression via data hiding. *Thirty-First Asilomar Conf. on Signals, Systems, and Computers*, pages 647–651, 1997.

Index

A

Absolute threshold
  definition, 533
  hearing, 273–274
Acrostic, historical perspective, 10
Active attack, definition, 42, 533
Active warden, definition, 428, 469, 533
Added mark, 88
Added pattern
  communication-based watermarking model, 67
  definition, 61, 533
Added vector/mark, definition, 533
Additive noise
  definition, 533
  valumetric distortion in watermark detection, 308–312
Additive white Gaussian noise (AWGN)
  channels, 155–156
  correlated Gaussian signals, 161–164
  definition, 534
  geometric interpretation of white Gaussian signals, 158–159
  lower bound on channel capacity, 170–172
  Shannon's theorem, 157, 159–161
Adversary
  capabilities
    attacker has detector, 347–348
    attacker knows algorithms, 347
    attacker knows nothing, 345–346
    collusion attack, 346–347
  definition, 23, 533
Alphabet, definition, 533
Ambiguity attack
  blind detection, 362–366
  countering, 366–367
  informed detection, 362
Amplitude change, valumetric distortion in watermark detection, 312–314
Approximate Gaussian method, normalized correlation analysis of false positive error, 240–242
Approximate restoration, 419
Array, notation, 62
Aspect ratio, definition, 26, 533
Asymmetric fingerprinting, definition, 533
Asymmetric-key cryptography, definition, 533
Attack
  categories
    system attacks, 345
    unauthorized detection, 340–341
    unauthorized embedding, 340
    unauthorized removal, 341–345
  definition, 533
  known attacks
    ambiguity attack
      blind detection, 362–366
      countering, 366–367
      informed detection, 362
    copy attack, 361–362
    gradient descent attacks, 372–373
    linear filtering attack, 360–361
    scrambling attack, 359
    sensitivity analysis attacks, 367–372
    synchronization attacks, 360
  localized authentication systems
    collage attacks, 415–419
    search attacks, 415
Attack channel, watermarking game, 512–513
Attacker *see* Adversary
Audio perceptual model
  masking, 274–276
  pooling, 276–277
  sensitivity, 273–274
Audio scaling, 326
Authentication, *see also* Exact authentication; Selective authentication
  definition, 534
  localization
    attacks
      collage attacks, 415–419
      search attacks, 415
    block-wise content authentication, 411–412
    overview, 410–411
    sample-wise content authentication, 412–414
  restoration
    approximate restoration, 419
    blind restoration, 421
    embedded redundancy, 419–420

exact restoration, 419
self-embedding, 420–421
Authentication mark
definition, 26, 534
modification, 27
Autocorrelation, temporal and geometric distortions, 329–330
AWGN, *see* Additive white Gaussian noise

B

Base bits, definition, 534
Basilar membrane spreading function, 275
BCH, *see* Bose-Chandhuri-Hocquenghem
Benchmarking, watermarking systems, 47–48
BER, *see* Bit error rate
Binary entropy function, matrix embedding, 455
Bit error rate (BER)
definition, 214, 534
modeling, 215–217
Blind coding, definition, 534
Blind detection, definition, 39, 534
Blind embedding
communication-based watermarking model
algorithm, 70–75
overview, 68
definition, 534
Blind restoration, 421
Blind steganalysis
definition, 51, 534
JPEG images using calibration, 486–489
overview, 473–474
spatial domain, 489–495
Blind watermarking, definition, 534
Block DCT
definition, 534
simulated JPEG compression, 321–322
Block-wise content authentication, 411–412
Bose-Chandhuri-Hocquenghem (BCH)
definition, 534
error correction code, 119
Brightness sensitivity
definition, 534
perceptual models, 265–266
Broadcast monitoring
active versus passive, 17–19
definition, 534
users, 16–17
watermarking, 16–19
Burst error
error correction codes, 119
message error, 214

C

Calibration, blind steganalysis of JPEG images, 486–489
Capacity, *see* Channel capacity; Embedding capacity; Information hiding
Caridi, Carmine, 25
Channel capacity
additive white Gaussian noise channel, 172–174
definition, 534
Channel decoder, definition, 534
Channel encoder, definition, 535
Chip, definition, 535
Cipher, definition, 536
Cipher key
definition, 535
overview, 43–45, 348–349
Ciphertext
definition, 535
overview, 348–349
secure transmission, 65
Cleartext
definition, 535
overview, 348–349
secure transmission, 65
Code book, definition, 535
Code-division multiplexing
definition, 106, 535
8-bit watermark system, 115–117
equivalence with other approaches, 113–114
multisymbol message coding, 112–113
Code separation
direct message coding, 107–110
performance effects, 187
Code vector, definition, 535
Code word, definition, 535
Collage attack, localized authentication system, 415–419
Collusion attack
definition, 535
watermark security, 42, 346–347
Collusion-secure code, definition, 535
Common functions, notation, 532
Common variables, notation, 529–531
Communication-based watermarking models
basic model, 67–75
multiplexed communications, 78–80
side information at transmitter, 75–78
Communications systems
components, 63–64

secure transmission, 65–67
transmission channel classes, 64–65
Compliant device, definition, 30, 535
Content
authentication, 25–27
definition, 535
Content scrambling system (CSS), definition, 536
Contrast sensitivity
contrast sensitivity function, 264–265
definition, 535
Convolution code, *see also* Trellis code
definition, 535
Copy attack
definition, 535
watermark, 361–362
Copy control
definition, 535
encryption, 30
watermarking, 27–31
Copy Protection Technical Working Group (CPTWG)
definition, 536
record control, 29
Copyright
notice, 19–20
protection, 19, 535
watermarks, 20–21
Correlation, *see* Correlation-based watermarking systems; Correlation coefficient; Linear correlation; Normalized correlation
Correlation-based watermarking systems
correlation coefficient
equivalent detection methods, 101–102
geometric interpretation, 100–101
linear correlation
equivalent detection methods, 96–97
geometric interpretation, 96
matched filtering, 96
normalized correlation
equivalent detection methods, 100
geometric interpretation, 98–99
overview, 95–96
Correlation coefficient
block-based, blind embedding, and correlation coefficient detection, 89–95
definition, 95, 100, 535
equivalent detection methods, 101–102
geometric interpretation, 100–101
Coset, definition, 535
Cover-<datatype>, definition, 536
Covert communications
definition, 536
example, 5
Cover Work, definition, 536
CPTWG, *see* Copy Protection Technical Working Group
Critical bandwidth
definition, 536
hearing, 274–275
Cryptography
definition, 536
digital content protection, 11
secure transmission, 65
watermark security relationship
overview, 348–349
unauthorized detection prevention, 349–351
unauthorized embedding prevention, 351
unauthorized removal prevention, 355–358
CSS, *see* Content scrambling system

D

DAC, *see* Data authentication code
Data authentication code (DAC), definition, 50
Data Hiding Sub-Group (DHSG), definition, 536
Data hiding, *see* Information hiding
Data payload
definition, 536
steganographic systems, 50
watermarking systems, 38
DCT, *see* Discrete cosine transform
Decipher, *see* Decryption
Decoder, *see* Watermark decoder
Decoding metric, definition, 185
Decorrelating filter, *see* Whitening filter
Decryption, definition, 536
DeCSS, definition, 536
Denoising, blind steganalysis in spatial domain, 491–494
Detection measure, definition, 83, 536
Detection region
definition, 80, 536
geometric watermarking models, 83–85
Detection statistic, definition, 536
Detector *see* Watermark detector
Device control
definition, 31, 536
watermarking, 31–32

DHSG, *see* Data Hiding Sub-Group
Difference expansion, erasable watermarks, 385–391
Digital rights management (DRM), definition, 536
Digital signature
  definition, 26, 536
  error correction code comparison, 419–420
Digital watermarking
  historical perspective, 8
  importance, 11–12
DIM, *see* Dithered index modulation
Direct binning, dirty-paper code design, 188
Direct message coding
  code separation, 107–110
  definition, 105, 537
  maximum likelihood detection, 107
Dirty-paper code
  collection of codes, 164
  Costa's insight, 170–175
  decoding algorithm efficiency, 185–186
  definition, 153, 537
  design approaches
    direct binning, 188
    dither modulation, 189
    quantization index modulation, 188–189
  encoding algorithm efficiency, 184–185
  Gaussian signal watermarking, 164–166, 168–169
  informed encoding/embedding using normalized correlation detection, 166–168
  lattice code
    dither modulation implementation with simple lattice code, 189–194
    E_8 lattice dirty-paper code construction, 201–204
    implementation tricks
      distortion compensation, 194–195
      dithering, 195–197
      lattice selection, 194
      spreading functions, 195
    lattice properties, 199–201
    nonorthogonal lattices, 197–199
    overview, 138
    valumetric scaling
      inverting valumetric scaling, 208
      overview, 204–205
      rational dither modulation, 207–208
      scale-invariant marking spaces, 205–207
  quincunx dirty-paper code, 180–181
  robustness versus encoding cost, 186–187
  trellis codes, 208–211
Discrete cosine transform (DCT), *see also* Watson's discrete cosine transform-based visual model
  definition, 537
  OutGuess and steganography, 440–441
  semi-fragile watermarks and quantizing discrete cosine transform coefficients, 399–404
Distortion compensation
  definition, 139
  Gaussian signal watermarking, 164–166
  lattice code implementation, 194–195
Distortion distribution
  definition, 81
  geometric watermarking models, 85–87
Distortion function, watermarking game, 511–512
Distortion quantization index modulation (DQIM)
  M-ary watermarking, 179
  performance, 178–181
  scalar watermarking, 175–178
Distribution of unwatermarked Works
  definition, 80, 537
  geometric watermarking models, 81–82
Dither modulation (DM)
  dirty-paper code design, 189
  implementation with simple lattice code, 189–194
  rational dither modulation, 207–208
Dithered index modulation (DIM), definition, 537
Dithered quantization, definition, 323, 537
DiVX watermark, 24
DM, *see* Dither modulation
Document, *see* Work
Document-to-watermark ratio (DWR), definition, 176, 537
DQIM, *see* Distortion quantization index modulation
DRM, *see* Digital rights management
DWR, *see* Document-to-watermark ratio

E

ECC, *see* Error correction code
Effectiveness, definition, 227, 537
Elimination attack
  definition, 537
  watermark, 42
Embedded-<datatype>, definition, 537

Embedder, definition, 3, 537
Embedding capacity
  definition, 537
  steganographic systems, 50, 433
Embedding distribution
  definition, 80, 85
  watermarking game, 513
Embedding efficiency
  definition, 53
  matrix embedding, 450
  steganographic systems, 49–51
  upper bound on embedding efficiency, 454–457
  watermarking systems, 37
Embedding modification, definition, 537
Embedding region
  definition, 80, 537
  geometric watermarking models, 85
Embedding strength, watermark embedders, 47
Encipher *see* Encryption
Encoder, definition, 537–538
Encryption, *see also* Cryptography
  copy control, 28, 30
  definition, 538
  thwarting, 28
Erasable watermark
  definition, 538
  exact authentication
    construction, 381–385
    design, 379–380
    difference expansion, 385–391
    erasability problems, 380
    histogram shifting, 392–395
Error correction code (ECC)
  burst errors, 119
  definition, 538
  digital signature comparison, 419–420
  examples
    BCH, 119
    Hamming code, 119
    trellis code
      decoding, 122–123
      8-bit watermarks, 123–124
      encoding, 120–122
      overview, 119
    turbo codes, 119
  principles, 118–119
  random errors, 119
  rationale, 117–118
Error correction decoding, definition, 185
Error correction encoder, definition, 538
Exact authentication
  definition, 376, 538
  embedded signatures, 378–379
  erasable watermarks
    construction, 381–385
    design, 379–380
    difference expansion, 385–391
    erasability problems, 380
    histogram shifting, 392–395
  fragile watermarks, 377–378
Exact restoration, 419
Extracted mark, definition, 88, 538
Extraction function, definition, 538

F

F5 algorithm, steganography, 448–449
Fading channel, definition, 538
False negative, definition, 538
False negative error
  definition, 213
  normalized correlation analysis
    overview, 250–251
    spherical method, 251–252
  rate, 225
False negative probability
  definition, 225–226, 538
  determinants, 226–227
False positive
  definition, 538
  steganographic systems, 53
  watermarking systems, 39–40
False positive error
  definition, 213
  difference evaluation between random-Work false positive and random watermark false positive behaviors, 221–225
  normalized correlation analysis
    approximate Gaussian method, 240–242
    spherical method, 243–246
    whitening filter algorithm, 247–250
  random watermark false positive, 219–221
  random-Work false positive, 219, 221
  sources, 218
False positive probability
  comparing against quantized vectors, 130
  definition, 40, 218, 538
  estimation using normalized correlation detector, 517–522
  individual symbol detection, 126–128
  multisymbol watermark detection, 126
False positive rate, definition, 40, 218, 538

Fidelity
  capacity with mean square error fidelity constraint, 514–517
  definition, 538
  steganographic systems, 50
  two alternative, forced choice analysis, 258–259
  watermark, 15, 256
  watermarking systems, 37–38
Fingerprinting, definition, 23, 538
Forensic steganalysis, overview, 475–476
Forgery, *see* Unauthorized embedding
Fragile watermark, definition, 26, 539
Frequency hopping
  definition, 539
  secure transmission, 66
Frequency sensitivity
  definition, 539
  perceptual models, 263–265
Frequency-division multiplexing
  definition, 106, 539
  multisymbol message coding, 111–112
Frequency-domain watermarking, definition, 539

G

Gaussian noise, *see* Additive white Gaussian noise
Gaussian signal watermarking with side information, *see also* Dirty-paper code; Distortion compensation
  correlated Gaussian signals, 161–164
  formal definition of problem, 153–155
  geometric interpretation of white Gaussian signals, 158–159
  multiple cover works, 157–58
  Shannon's theorem, 157, 159–161
  signal and channel models, 155–156
  single cover work, 156–157
Generational copy control, definition, 539
Geometric watermarking models
  block-based, blind embedding, and correlation coefficient detection, 89–95
  detection region, 83–85
  distortion distribution, 85–87
  distribution of unwatermarked works, 81–82
  embedding distribution or region, 85
  marking spaces, 87–89
  overview, 80–81
  region of unacceptable fidelity, 82–83
Gold sequences, definition, 539
Golden ear, definition, 258, 539
Golden eyes, definition, 258, 539
Gradient descent attacks, watermark, 372–373

H

Hamming code
  definition, 539
  error correction code, 119
  matrix embedding, 452–454
Hash function
  definition, 539
  one-way hash function, 508–510
Hearing, *see also* Audio perceptual model
  absolute threshold, 273–274
  frequency response, 263–264
  loudness sensitivity, 265–266
Histogram attack
  definition, 539
  least significant bit embedding, 478–480
Histogram characteristic function, definition, 539
Histogram shifting, erasable watermarks, 392–395

I

Illegitimate distortion
  definition, 395, 539
  overview, 395–398
Imperceptible, definition, 539
Implicit synchronization
  definition, 539
  overview, 331–332
Information hiding
  categories of systems, 5–6
  definition, 540
  game subject to distortions, 513
  overview, 4
  watermarking game and data-hiding capacity
    capacity with mean square error fidelity constraint, 514–517
    general capacity, 513–514
Information theory
  channel capacity
    additive white Gaussian noise channel, 501–503
    arbitrary channels, 501
    definition, 500–501

coding theory
covering radius, 503–504
Hamming distance, 503
communication rates
achievable rate, 500
code rate, 499–500
cryptographic signatures, 510
cryptography
applications, 505
asymmetric-key cryptography, 506–508
definition, 505
symmetric-key cryptography, 505–506
entropy, 497–498
linear codes
cosets, 504–505
generator matrix, 504
parity matrix, 504
syndromes, 504
mutual information, 498–499
one-way hash functions, 508–510
steganography, 433–439
watermarking game analysis, *see* Watermarking game
Informed coding, definition, 138, 540
Informed detection, definition, 39, 51, 137, 540
Informed embedding
communication-based watermarking model, 68
definition, 137, 540
embedding as optimization problem, 140–141
fragility of fixed normalized correlation strategy, 144–146
optimization with respect to detection statistic, 141–144
optimizing with respect to robustness estimate, 137–152
watermarking, 137–138
Inseparability, definition, 540
International Telecommunications Union (ITU), definition, 540
Invariant watermark, properties, 330
Invertibility, ambiguity attacks, 367
Invertible watermark, definition, 540
ITU, *see* International Telecommunications Union

J

Jamming, definition, 540
JND, *see* Just noticeable difference
JPEG, definition, 540
Just noticeable difference (JND)
definition, 258, 540
region of acceptable fidelity, 83
Watson's discrete cosine transform-based visual model, 270–273

K

Kerckoff's assumption, 347
Key, *see* Cipher key; Watermark key
Key management, definition, 540
Keyspace, definition, 540
Kissing number, lattice, 200
Known cover attack, 476
Known message attack, 476
Known stego method attack, 476

L

Lattice code
definition, 540
overview, 179
dither modulation implementation with simple lattice code, 189–194
implementation tricks
distortion compensation, 194–195
dithering, 195–197
lattice selection, 194
spreading functions, 195
nonorthogonal lattices, 197–199
lattice properties, 199–201
E_8 lattice dirty-paper code construction, 201–204
valumetric scaling
inverting valumetric scaling, 208
overview, 204–205
rational dither modulation, 207–208
scale-invariant marking spaces, 205–207
Least significant bit (LSB) embedding
matrix embedding, 450, 456
definition, 540
histogram attack, 478–480
Least significant bit watermarking, 175
Legacy enhancement, watermarking, 32–33
Legitimate distortion
definition, 395, 540
overview, 395–398
Letterbox, definition, 541
Linear correlation
definition, 95, 541
equivalent detection methods, 96–97
geometric interpretation, 96
matched filtering, 96

optimal perceptual shaping, 287–288
quantization noise effects on watermark detection, 323–325
whitened linear correlation detection, 234–239
Linear filtering, valumetric distortion in watermark detection, 314–319
Linear filtering attack, watermark, 360–361
Localization, *see* Authentication
Lossy compression
legitimate distortion, 396
valumetric distortion in watermark detection, 319
Loudness sensitivity
definition, 541
perceptual models, 265–266
Lower embedding efficiency, matrix embedding, 450–451
L_p-norm, definition, 541
LSB embedding, *see* Least significant bit embedding

M

MAC, *see* Message authentication code
Malicious warden, definition, 428, 469, 541
Marking space
abstract view, 154
definition, 541
geometric watermarking models, 87–89
scale-invariant marking spaces, 205–207
Masking
audio perceptual model, 274–276
definition, 541
perceptual models, 266–268
Watson's discrete cosine transform-based visual model
contrast, 271
luminance, 270–271
Masking attack
definition, 541
watermark, 42
Matched filtering, 96
Matrix embedding, definition, 541
Matrix LT process, nonshared selection rule, 460–464
Maximum likelihood detection, direct message coding, 107
Mean squared error (MSE)
capacity with mean squared error fidelity constraint, 514–517
perceptual model limitations, 261–263
region of acceptable fidelity, 82–83
Media, definition, 2, 541
Media space
abstract view, 154
definition, 81, 541
distributions and regions, 81–87
MediaSync, 33
Message, definition, 541
Message authentication code (MAC), definition, 509, 541
Message bits, *see* Base bits
Message coding, *see* Direct message coding; Error correction code; Multisymbol message coding
Message error
bit error, 214
bit error rate modeling, 215–217
burst error, 214
definition, 213
Message mark, definition, 105, 541
Message pattern
communication-based watermarking model, 68
definition, 541
Messaging layer, definition, 542
Metric, definition, 542
Minkowski summation, *see* L_p-norm
Model-based steganography, definition, 542
Modulator, definition, 542
Moment-generating function, quantization noise effects on watermarks, 525–526
Mosaic attack, watermark, 359
MP3, definition, 542
MPEG, definition, 542
MSE, *see* Mean squared error
m-sequence, definition, 541
Multimedia object, *see* Work
Multiplexed communications, watermarking as, 78–80
Multisymbol message coding
code-division multiplexing, 112–114
definition, 542
8-bit watermark system, 115–117
frequency-division multiplexing, 111–112
overview, 110–111
space-division multiplexing, 111
time-division multiplexing, 111
Multisymbol watermark detection
comparing against quantized vectors
application, 129–130
block-based 8-bit watermarking with normalized correlation detection, 131–133

false positive probability, 130
overview, 128–129
individual symbol detection
application, 126
false positive probability, 126–128
looking for valid messages
application, 125–126
false positive probability, 126
overview, 124–125
Mutual information, Gaussian signals, 158

N

National Television Systems Committee (NTSC), definition, 542
Natural robustness, 327
N-ball, definition, 542
N-bit watermark, definition, 38
N-cone, definition, 542
Noncompliant device, definition, 30
Nonshared selection rule
matrix LT process, 460–464
overview, 457–458
perturbed quantization, 464–467
wet-paper codes with syndrome coding, 458–459
Nonsubtractive dither
definition, 542
dithered quantization, 323
Normalized correlation
definition, 95, 98, 542
equivalent detection methods, 100
error analysis
false negative error
overview, 250–251
spherical method, 251–252
false positive error
approximate Gaussian method, 240–242
spherical method, 243–246
whitening filter algorithm, 247–250
fixed robustness embedding for normalized correlation detector, 149–152
fragility of fixed normalized correlation strategy in informed embedding, 144–146
geometric interpretation, 98–99
informed encoding/embedding using normalized correlation detection, 166–168
optimal perceptual shaping, 291–294
N-sphere, definition, 542
NTSC, *see* National Television Systems Committee

O

Oblivious, definition, 542
1 embedding, definition, 533
One-way hash, definition, 542–543
Operators, notation, 530
OutGuess, steganography, 440–441

P

Panscan, definition, 543
Parity bits, definition, 543
Passive attack, *see* Unauthorized detection
Passive warden, definition, 428, 469, 543
Patchwork, definition, 543
Payload, *see* Data payload
Perceptual models, *see also* Audio perceptual model; Mean squared error; Perceptually adaptive watermarking; Two alternative, forced choice; Watson's discrete cosine transform-based visual model
masking, 266–268
optimal use
fixed fidelity, 288–289
fixed linear correlation, 289–290
perceptual shaping
linear correlation system, 287–288
normalized correlation system, 291–294
scaled embedding, 290–291
pooling, 267, 269
sensitivity
frequency sensitivity, 263–265
loudness/brightness sensitivity, 265–266
Perceptual shaping, definition, 543
Perceptual slack, definition, 543
Perceptually adaptive watermarking
definition, 543
embedding
perceptually limited embedding, 277–280
perceptually shaped embedding, 284–287
overview, 277
perceptual shaping
optimal perceptual shaping
linear correlation system, 287–288
normalized correlation system, 291–294
overview, 280–284
Perceptually insignificant, definition, 301

Perceptually significant
components, 543
embedding in perceptually significant coefficients, 301–302
Perturbed quantization, nonshared selection rule, 464–467
Perturbed quantization steganography, definition, 543
Pirate, definition, 23, 543
Playback control, definition, 30, 543
PN pattern, *see* Pseudo-random noise pattern
Pooling
audio perceptual model, 276–277
definition, 543
perceptual models, 268, 270
Watson's discrete cosine transform-based visual model, 271, 273
Power-spectrum condition, definition, 543
Precancellation, 140
Predistorted watermark, *see* Robust watermark
Prisoners' problem, definition, 543
Private key, asymmetric-key cryptography, 507
Private watermarking
definition, 543
overview, 338–340
systems, 39
Probability density function
quantization noise effects on watermarks, 524–525
random variables, 529
Projected mark, *see* Extracted mark
Proof of ownership, definition, 37, 543
Pseudo-random noise (PN) pattern, watermarking, 45
Public key, asymmetric-key cryptography, 507
Public-key steganography, definition, 544
Public watermarking
definition, 544
overview, 338–340
systems, 39

Q

QIM, *see* Quantization index modulation
Quality
change, 256
definition, 544
improvement, 256
two alternative, forced choice analysis, 260
watermark, 256
Quantization
dithered quantization, 323
factor, 320
modeling, 320
noise
characteristics, 320–321
linear correlation watermark detection effects, 323–325
watermark effect analysis
expected correlation for Gaussian watermark and Laplacian content, 527–528
moment-generating function, 525–526
overview, 522–524
probability density function, 524–525
perturbed quantization, 464–467
semi-fragile watermarks and quantizing discrete cosine transform coefficients, 399–404
simulated JPEG compression, 321–322
Quantization index modulation (QIM)
definition, 544
dirty-paper code design, 188–189
scalar watermarking, 174–175
Quantization watermarking, definition, 544
Quincunx dirty-paper code, 179–180

R

Random errors, error correction codes, 119
Random scalar variable, notation, 62
Random vector, notation, 62–63
Random watermark false positive
difference evaluation between random-Work false positive and random watermark false positive behaviors, 221–225
overview, 219–221
Random watermark false positive probability
definition, 219, 544
whitening filter effects, 234
Random-Work false positive
difference evaluation between random-Work false positive and random watermark false positive behaviors, 221–225
overview, 219, 221
Random Work false positive probability, definition, 219, 544
Receiver operating characteristic (ROC) curve
hypothetical curve, 228–230
interpolation, 231

real system curve, 228–231
watermarking system evaluation, 228
Record control
definition, 29, 544
whitening filter effects, 234
Redundant embedding, robust watermark construction, 299–300
Reference mark, definition, 88, 544
Reference pattern
communication-based watermarking model, 67
definition, 61, 544
Reference vector, *see* Reference mark
Region of acceptable distortion, *see* Region of acceptable fidelity
Region of acceptable fidelity
definition, 80, 544
geometric watermarking models, 82–83
Registration, *see also* Synchronization
blind detectors, 328–329
Relative length, definition, 433
Removal attacks, *see* Unauthorized removal
Restoration, *see* Authentication
Robustness
definition, 227, 544
encoding cost tradeoff in dirty-paper code, 186–187
informed embedding optimizing with respect to robustness estimate, 137–152
natural robustness, 327
steganographic systems, 53
temporal and geometric distortions
autocorrelation, 329–330
exhaustive search after distortion, 327–328
implicit synchronization, 331–332
invariant watermarks, 330
overview, 325–326
synchronization/registration in blind detectors, 328–329
watermark, 15, 40–41, 297
Robust watermark
construction
embedding in coefficients of known robustness, 302–303
embedding in perceptually significant coefficients, 301–302
inverting distortions in detector, 303–304
overview, 298–299
preinverting distortions in embedder, 304–308
redundant embedding, 299–300
spread spectrum coding, 300–301
secure watermark comparison, 297
valumetric distortion in watermark detection
additive noise, 308–312
amplitude change, 312–314
linear filtering, 314–319
lossy compression, 319
quantization, 320–325
ROC curve, *see* Receiver operating characteristic curve

S

Sample pairs analysis
definition, 544
steganalysis, 480–486
Sample-wise content authentication, 412–414
Scalar Costa scheme, *see* Distortion quantization index modulation
Scalar watermarking
distortion quantization index modulation, 175–178
quantization index modulation, 174–175
Scalar, notation, 62
Scrambling attack, watermark, 359
SDMI, *see* Secure Digital Music Initiative
Search attack, localized authentication system, 415
Second-generation watermarking, definition, 545
Secure Digital Music Initiative (SDMI)
copy control, 29
definition, 544
Secure transmission, definition, 545
Security
attack types, 41–43
definition, 41, 227, 545
steganographic systems, 54
watermark, *see* Watermark security
Selection rule
definition, 545
types, 431–432
Selective authentication
definition, 376, 545
legitimate versus illegitimate distortion, 395–398
overview, 395
semi-fragile signatures, 404–409
semi-fragile watermarks
design, 399

quantizing discrete cosine transform coefficients, 399–404
telltale watermarks, 409–410
Self-embedding, 420–421
Semi-fragile signature
embedded with semi-fragile watermarks, 405–409
selective authentication, 404–405
Semi-fragile watermark
definition, 27, 545
design, 399
embedded with semi-fragile signatures, 405–409
quantizing discrete cosine transform coefficients, 399–404
Sensitivity
audio perceptual model, 273–274
definition, 545
perceptual models
frequency sensitivity, 263–265
loudness/brightness sensitivity, 265–266
Watson's discrete cosine transform-based visual model, 270
Sensitivity analysis attack
definition, 545
watermark, 367–372
Set, notation, 62
Shannon's theorem, additive white Gaussian noise channel, 157, 159–161
Shaped pattern, definition, 545
Side information
definition, 545
transmitter side information and watermarking, 75–78
watermarking, *see also* Dirty-paper code; Distortion compensation
correlated Gaussian signals, 161–164
formal definition of problem, 153–155
geometric interpretation of white Gaussian signals, 158–159
lattice codes, 179–180
multiple cover works, 157–168
scalar watermarking
distortion quantization index modulation, 175–182
quantization index modulation, 175–176
Shannon's theorem, 157, 159–161
signal and channel models, 155–156
single cover work, 156–157
Signal-to-noise ratio (SNR), definition, 545
Signal-to-watermark ratio, definition, 545
Signature, *see* Digital signature
Slack, *see* Perceptual slack
definition, 545
SNR, *see* Signal-to-noise ratio
Soft decoding scheme, definition, 186
Sound pressure level (SPL), definition, 273, 545
Space-division multiplexing
definition, 106, 545
multisymbol message coding, 111
SpamMimic, stego Work production, 431
Spectral flatness measure, 275
Sphere packing
density in lattice, 199–200
performance effects, 187
Spherical method
false positive probability estimation using normalized correlation detector, 517–522
normalized correlation error analysis
false negative error, 251–252
false positive error, 243–246
SPL, *see* Sound pressure level
Spreading function, lattice code implementation, 195
Spread spectrum, definition, 545
Spread spectrum coding, robust watermark construction, 300–301
Spread spectrum communications
definition, 44, 546
secure transmission, 66
Steganalysis
algorithms
blind steganalysis
JPEG images using calibration, 486–489
spatial domain, 489–494
databases for testing, 477
least significant bit embedding and histogram attack, 478–480
sample pairs analysis, 480–486
cover Work influence, 476–477
definition, 426, 546
detection
blind steganalysis, 473–474
overview, 470–472
system attacks, 474–475
targeted steganalysis, 472–473
forensic steganalysis, 475–476
statistical undetectability, 52–53
Steganographic capacity, definition, 16, 50, 546

Steganographic message, definition, 546
Steganographic security
  Cachin's definition, 434–439, 534
  definition, 546
Steganographic systems
  channel
    active warden, 428
    malicious warden, 428
    passive warden, 428
  design building blocks, 429–432
  evaluation, 55–56
  properties
    blind extraction, 51
    data payload, 50
    embedding effectiveness, 49–50
    false alarm rate, 53
    fidelity, 50
    informed extraction, 51
    overview, 49
    robustness, 53
    security, 54
    steganographic capacity, 50
    targeted steganalysis, 51
Steganography
  applications
    criminals, 35–36
    dissidents, 34–35
    overview, 34
  attributes, 16
  definition, 1, 425, 546
  embedding impact minimization
    matrix embedding
      embedding efficiency, 450
      Hamming codes, 452–454
      least significant bits embedding, 450, 456
      lower embedding efficiency, 450–451
      upper bound on embedding efficiency, 454–457
    nonshared selection rule
      matrix LT process, 460–464
      overview, 457–458
      perturbed quantization, 464–467
      wet-paper codes with syndrome coding, 458–459
    overview, 449
  embedding scheme, 427
  historical perspective, 3–4, 9–11
  importance, 12–13
  information theory, 433–439
  notation and terminology, 433
  overview, 425–427
  practical methods
    masking embedding as natural processing
      F5 algorithm, 448–449
      stochastic modulation, 445–448
    model-based steganography, 441–444
    statistics preserving steganography, 439–441
  versus watermarking, 4
Steganology, definition, 2, 546
Stego-<datatype>, definition, 546
Stego key
  definition, 546
  design, 432
  overview, 54–55
Stirmark program, 48, 546
Stochastic modulation
  definition, 546
  masking embedding as natural processing, 445–448
Structured codes, definition, 546
Subtractive dither
  definition, 546
  dithered quantization, 323
Symbol assignment function, definition, 546
Symbols, definition, 546
Symmetric-key cryptography, definition, 546
Synchronization
  blind detectors, 328–329
  definition, 546
  implict synchronization, 331–332
Synchronization attack, watermark, 360
Syndrome
  definition, 165
  definition, 546
  mapping, 179
  matrix embedding, 451
  parameter, 170
  quantization, 170
  wet-paper codes with syndrome coding, 458–459
Syndrome code, definition, 546
System attack
  definition, 546
  steganographic algorithms, 474–475
  watermark security, 345

T

Tamper resistance, definition, 547
Targeted steganalysis
  definition, 51, 547
  overview, 472–473
Telecine process, definition, 547

Telltale watermark
  definition, 547
  selective authentication, 409–410
Time-division multiplexing
  definition, 106, 547
  multisymbol message coding, 111
Traitor, definition, 23, 547
Transaction tracking
  definition, 23, 547
  watermarking, 23–25
Transparent watermark, definition, 547
Transport layer, definition, 547
Trellis code
  decoding, 122–123
  definition, 547
  dirty-paper trellis code, 208–211
  8-bit watermarks, 123–124
  encoding, 120–122
  overview, 119
Trellis-coded modulation, definition, 547
Trustworthy camera, definition, 547
Turbo code
  definition, 547
  error correction codes, 119
2AFC, *see* Two alternative, forced choice
Two alternative, forced choice (2AFC)
  definition, 547
  fidelity assessment, 258–259
  overview, 258
  quality assessment, 260

U

Unauthorized deletion, *see* Unauthorized removal
Unauthorized detection
  definition, 547
  passive attack, 42–43, 337
  prevention, 349–351
  watermark security, 340–341
Unauthorized embedding
  definition, 547
  forgery of watermarks, 43, 337
  prevention, 351
  watermark security, 340
Unauthorized reading, *see* Unauthorized detection
Unauthorized removal
  definition, 547
  elimination attack, 42
  masking attack, 42
  prevention, 355–358
  watermark security, 337, 341–345
Unauthorized writing, *see* Unauthorized embedding
Undetectability
  definition, 548
  steganography, 425–426
Unobtrusive, *see* Imperceptible

V

Valumetric distortion
  definition, 548
  watermark detection
    additive noise, 308–312
    amplitude change, 312–314
    linear filtering, 314–319
    lossy compression, 319
    quantization, 320–325
Valumetric scaling
  inverting valumetric scaling, 208
  overview, 204–205
  rational dither modulation, 207–208
  scale-invariant marking spaces, 205–207
VBI, *see* Vertical blanking interval
Vector, notation, 62, 529
VEIL, video watermarking, 31
Vertical blanking interval (VBI), definition, 548
Visible watermark, definition, 548
Vision, *see also* Watson's discrete cosine transform-based visual model
  frequency response, 263–264
  perceptual models, *see* Perceptual models
Viterbi decoder, definition, 548
Voronoi cell, lattice, 200

W

Warden, types, 428, 469–470
Watermark
  applications
    broadcasting, 16–19
    content authentication, 25–27
    copy control, 27–31
    device control, 31–32
    legacy enhancement, 32–33
    owner identification, 19–21
    proof of ownership, 21–23
    transaction tracking, 23–25
  attributes, 1, 15
  definition, 61, 548
  modification, 45
  multiple, 45–46

Watermark decoder
communication-based watermarking model, 69
definition, 548
Watermark detector
definition, 3, 548
detection threshold, 47
embedding steps, 87–89
Watermark extraction, 87
Watermarking
costs, 46
definition, 2, 548
historical perspective, 6–9
models, *see* Communication-based watermarking models; Correlation-based watermarking systems; Geometric watermarking models
versus steganography, 4
Watermarking game
attack channel, 512–513
data-hiding capacity
general capacity, 513–514
capacity with mean square error fidelity constraint, 514–517
distortion function, 511–512
embedding distribution, 513
Watermarking systems
abstract view, 153–155
evaluation
benchmarking, 47–48
performance parameters, 47
scope of testing, 48–49
properties
blind detection, 39
data payload, 38–39
embedding effectiveness, 37
false positive, 39–40
fidelity, 37–38
informed detection, 39
overview, 36–37
robustness, 40–41
security, 41–43
Watermark key
definition, 548
overview, 43–45
Watermark-to-noise ratio (WNR), definition, 176, 548
Watermark pattern, definition, 548
Watermark security
adversary capabilities
attacker has detector, 347–348
attacker knows algorithms, 347
attacker knows nothing, 345–346
collusion attack, 346–347
attack categories
system attacks, 345
unauthorized detection, 340–341
unauthorized embedding, 340
unauthorized removal, 341–345
cryptography relationship
overview, 348–349
unauthorized detection prevention, 349–351
unauthorized embedding prevention, 351
unauthorized removal prevention, 355–358
known attacks
ambiguity attack
blind detection, 362–366
countering, 366–367
informed detection, 362
copy attack, 361–362
gradient descent attacks, 372–373
linear filtering attack, 360–361
scrambling attack, 359
sensitivity analysis attacks, 367–372
synchronization attacks, 360
requirements
overview, 335–336
public and private watermarking, 338–340
restricting watermark operations, 336–338
Watermark vector, definition, 548
Watson's discrete cosine transform-based visual model
masking
contrast, 271
luminance, 270–271
overview, 270
pooling, 271, 273
sensitivity, 270
Wet-paper code
definition, 548
syndrome coding, 458–459
Whitening filter
decorrelation, 223
design, 232
error rate effects, 232–234
linear correlation detection, 234–239
normalized correlation detection of false positives, 247–250
WNR, *see* Watermark-to-noise ratio
Work, definition, 2, 548

This page intentionally left blank

About the Authors

Ingemar J. Cox is currently Professor and BT Chair of Communications in the Departments of Computer Science, and Electronic and Electrical Engineering at University College London and Director of UCL's Adastral Park Postgraduate Campus. Dr. Cox is a holder of a Royal Society Wolfson Fellowship. He received his B.Sc. from University College London and Ph.D. from Oxford University. He was a member of the Technical Staff at AT&T Bell Labs at Murray Hill from 1984 until 1989, where his research interests were focused on mobile robots. In 1989, he joined NEC Research Institute in Princeton, NJ, as a senior research scientist in the computer science division. While at NEC, his research shifted to problems in computer vision and he was responsible for creating the computer vision group at NECI. He has worked on problems to do with stereo and motion correspondence and multimedia issues of image database retrieval and watermarking. In 1999, Dr. Cox was awarded the IEEE Signal Processing Society Best Paper Award (Image and Multidimensional Signal Processing Area) for a paper he coauthored on watermarking. From 1997–1999, he served as Chief Technical Officer of Signafy, Inc., a subsidiary of NEC responsible for the commercialization of watermarking. Between 1996 and 1999, he led the design of NEC's watermarking proposal for DVD video discs and later collaborated with IBM in developing the technology behind the joint "Galaxy" proposal supported by Hitachi, IBM, NEC, Pioneer, and Sony. In 1999, he returned to NEC Research Institute as a Research Fellow. Dr. Cox is a Fellow of the IEEE, a Fellow of the IEE, and a Fellow of the Royal Society for Arts and Manufactures. He was founding coeditor in chief of the IEE Proceedings on Information Security and is an associate editor of the IEEE Transactions on Information Forensics and Security. He is coauthor of a book entitled *Digital Watermarking* and the coeditor of two books, *Autonomous Robots Vehicles* and *Partitioning Data Sets: With Applications to Psychology, Computer Vision, and Target Tracking*.

Matthew L. Miller began working in graphics and image processing at AT&T Bell Labs in 1979, and obtained a B.A. in cognitive science from the University of Rochester in 1986. Since then he has written several commercial software applications and delivered lecture courses at a number of universities in Europe. From 1993–1997, he divided his time between running Baltic Images, a company he founded in Lithuania, and consulting for NEC Research Institute. Between 1997 and 1999, he was employed by Signafy, Inc., where he was jointly responsible for the design, implementation, and testing of state-of-the-art video watermarking technologies. He was also a key participant of NEC's and later Galaxy's DVD watermarking proposals. He is currently a scientist at NEC Research Institute.

Jeffrey A. Bloom manages content security research in the United States for Thomson Corporate Research and oversees its worldwide digital watermarking research, coordinating watermarking research efforts across the company's facilities in Europe, Asia, and North America. Currently based at Thomson's Princeton Corporate Research Center, Dr. Bloom came to the global technology corporation from the Sarnoff Corporation, where he was technology leader from 2000–2005. As founder of Sarnoff's Information Security and Embedded Communications Laboratory, he oversaw the development of a digital watermarking solution for digital cinema consulted to the United States government on watermarking and steganography and developed information security techniques based on detection of anomalous content usage. Dr. Bloom began his career in 1998 as a research engineer at NEC Research Institute, where he was part of a team that developed watermarking technologies and applications for copy protection, authentication, and forensic tracking of both still and moving imagery. He is a member of the IEEE and the ACM, associate editor of *Springer LNCS Transactions on Data Hiding and Multimedia Security* and *EURASIP Journal on Information Security*, and is a reviewer for multiple journals and conferences. He holds a Ph.D. and M.S. in Electrical and Computer Engineering, the former from University of California, Davis and the latter from Worcester Polytechnic Institute, where he also earned his B.S.

Jessica Fridrich has been working in the field of digital watermarking and steganography since 1996. She is Professor at the Department of Electrical and Computer Engineering at Binghamton University (SUNY) where she teaches courses in steganography and digital forensics. She has received her Ph.D. in Systems Science from Binghamton University in 1995 and M.S. in Applied Mathematics from Czech Technical University in Prague in 1987. Her main interests are in steganography, steganalysis, digital watermarking, and digital image forensic. Dr. Fridrich's research work has been generously supported by the U.S. Air Force. Since 1995, she received 18 research grants totaling over $5 million for projects on data embedding and steganalysis that lead to more than 80 papers and 7 U.S. patents. Dr. Fridrich is a member of IEEE and ACM.

Ton Kalker is a senior researcher at Hewlett-Packard Laboratories and a faculty member of the Technical University of Eindhoven. His primary responsibility is to support Hewlett-Packard in the area of media security, including interoperability of digital rights management, watermarking, robust hashing, and advanced encryption technologies. Dr. Kalker received his Ph.D. from the University of Leiden in 1986. From 1983–1985, he was a faculty member of the University of Delft. From 1985–1994, he was a member of the staff at Philips Research in Eindhoven, where he developed widely deployed watermarking and fingerprinting solutions. Dr. Kalker is a Fellow of the IEEE for his contributions to practical applications of watermarking, in particular watermarking for DVD-Video copy protection. He cofounded the IEEE Transactions on Information

Forensics and Security. He has published extensively on media security–related topics and holds numerous patents in this field. He is an often invited speaker at conferences and consults for a number of major media security–related international projects. Currently, he is one of the leading architects of the Coral consortium on DRM interoperability.

This page intentionally left blank